⚡🚀Mike's Daily Paper, 22.11.2024: ⚡🚀
The Unreasonable Ineffectiveness of the Deeper Layers

📑 **Paper Review:** A Simple Approach to Transformer Layer Pruning

This weekend's paper presents a quite intuitive method for pruning layers in Transformer-based models. As you probably remember, the modern language models and models in other domains (like Computer Vision and Audio) are based on Transformers, which consist of blocks(layers) made up of an attention mechanism and two feed-forward layers (the second one being linear). These also include normalization layers and residual connections (where each layer's output is combined with the previous layer's output).

🔍 **Key Problem:** Modern language models contain dozens of Transformer layers, significantly impacting computation time and resources, especially in generation tasks. This paper proposes a method to prune consecutive transformer layers to reduce computation time. The challenge is selecting which blocks to remove while minimizing accuracy loss.

📋 **Methodology:** Given that the Transformer's computational graph contains multiple residual connections, it's natural to select consecutive blocks that don't contribute significantly to their preceding block's output. In other words, if these blocks' output delta is negligible, they can be removed without seriously impacting performance.

📊 **Implementation:** The authors chose to compare the output of the consecutive transformer layers from layer l+1 to the layer l+n (where n is the number of consecutive blocks to be removed) using a modified cosine distance metric (replacing cos with arccos and dividing by pi to normalize the metric between 0 and 1). Logically, n blocks showing high similarity to their preceding block's output are considered good candidates for pruning. The similarity is computed on the last token representation across a large dataset.

🔧 **Results:** After pruning, the model can undergo light fine-tuning. The authors claim this method can remove up to half of the Transformer layers (in language models) without significant performance degradation.

💡 **Significance**: This approach offers a computationally efficient way to reduce model size while maintaining performance, particularly valuable for deployment in resource-constrained environments.

https://arxiv.org/abs/2403.17887

⚡🚀Mike's Daily Paper, 23.11.2024: ⚡🚀
Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study

Today I'm reviewing a paper on a topic I haven't covered in a while - tabular data. The paper examines a fascinating question - do Large Language Models (LLMs) like GPT truly understand structured information in tables?

Background
In recent years, LLMs have become an important tool in natural language processing. But while they excel (sort of) at understanding natural language (in text form), their ability to understand information in tabular form hasn't been thoroughly researched, and that's exactly what the researchers are trying to do in this reviewed paper.

What did the researchers do?
The researchers developed a new metric called SUC (Structural Understanding Capabilities) that examines models' abilities to understand table structure. The metric includes seven different tasks:
- Table boundary identification
- Specific cell location
- Reverse lookup (location to value)
- Column retrieval
- Row retrieval
- Table size detection
- Merged cells identification

They tested GPT-3.5 and GPT-4 on these tasks using different input formats (HTML, JSON, CSV, and more).

What did they find?
The results are surprising! Here are the key points:
- HTML format emerges as the most "comfortable" format for presenting tables to LLMs
- The models showed good capabilities in relatively complex tasks (table boundary identification, merged cells identification) but failed at simple tasks (table size detection, simple row retrieval, single cell search)
- Performance improved significantly with one example (one-shot) compared to zero examples

The Main Innovation: Self-augmented Prompting
The researchers developed a new method called "self-augmented prompting" that improves model performance. The method first asks the model to identify critical information in the table (like value ranges) and then uses this information to improve the final answer. This enables quite significant improvement in several benchmarks.

Summary
I must say this paper is fascinating. It shows that despite the enormous progress in LLMs, there are still significant gaps in their ability to understand structured information. It reminds us that although these models are impressive, they're still far from true human understanding of structures and relationships between data.

The researchers did a good job developing metrics and methods that will help the community continue to improve these capabilities. Their new prompting method is simple but effective, and that's exactly what we need - practical solutions that can be implemented immediately.

Final Word
If you work with tables and LLMs, this paper is a must-read. It provides practical insights and useful tools. The code and data are available on GitHub, so you can start playing with it right away.

It will be particularly interesting to see how these findings influence the next generation of language models. Will we see models specifically designed for understanding structured information?

⚡🚀Mike's Daily Paper, 27.11.2024: ⚡🚀
The Illusion of State in State-Space Models

This important paper examines the theoretical limitations of State Space Models (SSMs), which emerged as an alternative architecture to transformers for LLMs. The authors demonstrate that despite their seemingly recurrent and stateful design, SSMs (like transformers) are fundamentally limited in their ability to express "continuous" computation, as they cannot compute anything outside the TC0 complexity class. TC0 tasks are defined as those that can be represented with basic Boolean circuits (and threshold and majority vote computations) at finite depth (e.g., addition of numbers, multiplication, or sorting of n numbers). This is the "simplest" class in the circuit complexity hierarchy.

This means that SSMs cannot solve permutation composition problems that single-layer RNNs are capable of solving.

Key Contributions of the Paper:

1. **Theoretical Analysis**

- Proves that both linear SSMs and Mamba-style SSMs are limited to TC0 computational complexity
- Shows that SSMs cannot solve NC1-complete problems (tasks that can be represented with Boolean operations at logarithmic depth from the problem dimension - number of variables in big-O) such as permutation composition. That is, not finite depth as in TC0.
- Demonstrates that SSMs cannot accurately track chess moves, write complex code, or track entities in narratives.

2. **Empirical Tests Conducted by the Paper:**

- Provides experimental evidence showing that Mamba-style SSMs and transformers struggle with permutation composition tasks
- Shows that SSMs require increasing depth to "handle" longer sequences for modeling "permutative" group operations

- Demonstrates that single-layer RNNs can solve these tasks that SSMs cannot (probably due to linearity between hidden state transfers in SSMs)

3. **Proposed Architectural Improvements:**

- Suggests 2 ways to extend SSMs beyond TC0 limitations: adding non-linearity (RNN-SSM) and making transition matrices input-dependent (WFA-SSM) - a refinement of Mamba that adds non-linearity to matrix A which remained constant in Mamba.

Impact and Implications of the Paper:

- Challenges assumptions about SSMs advantages over transformers
- Points to potential hybrid approaches combining different architectures
- Opens new directions for developing architectures with improved expressiveness for natural language processing applications and additional domains
- Emphasizes the importance of theoretical analysis of model architecture suitability for specific tasks it's designed to solve

Summary:
The paper contributes both theoretically and practically to understanding neural network architectures. The rigorous theoretical analysis, combined with supporting empirical evidence, provides important insights regarding the fundamental limitations of SSMs. While some theoretical results rely on complexity theoretical assumptions, the practical implications are well supported by empirical evidence.

https://arxiv.org/abs/2404.08819

⚡🚀 Mike's Daily Paper - 28.11.24: ⚡🚀
Parameter-Efficient Fine-Tuning with Discrete Fourier Transform

Background: PeFT
Let's start with a quick refresh about parameter-efficient fine-tuning methods. PeFT is a family of methods that enable fine-tuning of large models (particularly language models) using a minimal number of parameters, significantly saving computational resources and memory.

Background: LoRA:
One of the most popular methods in PeFT, called LoRA, freezes the model weights and trains additional(=delta) matrices for each transformer layer. Each learned additional matrix is low-rank, meaning it can be represented by multiplying two smaller matrices (in the middle dimension of the multiplication).

The main advantage of LoRA is that it allows adapting large models to specific tasks while training only a small portion (say 1% of all parameters), making it particularly efficient. This method has proven especially effective in adapting large language models to specific tasks. Additionally, LoRA enables quick switching between different versions of the fine-tuned model since the small matrices can be stored separately from the original model.

Proposed Method:

The central idea is to view the neural network weight changes like an image or signal, and represent them in frequency domain instead of direct values. When we want to fine-tune the model, instead of changing all weights directly (which requires many parameters), we:

1. Predefine sampling points in frequency space where we want to focus. This is like choosing which frequencies we want to keep in compressed delta matrix representation. This is done by selecting a fixed (non-learned) frequency matrix E of size 2xn used to construct the delta matrix representation. This matrix is constant for all transformer layers.
2. Learn a vector c of size n (for each layer) which, when combined with E, builds the additional matrix in frequency domain F (the exact construction method isn't clear in the paper)
3. Pass F through a Gaussian bandpass filter (mainly sampling low frequencies near the matrix center).
4. Transform matrix F to time (regular) domain and use it just like in LoRA

Advantages of the Proposed Method:
The major advantage is that frequencies are a very efficient way to represent information (requiring 2n + Ln weights where L is the number of model layers). Just like we can compress images or music by keeping the most important frequencies, here we can represent complex weight changes using a very small number of frequencies.

This works well (presumably) because:

- Weight changes tend to be relatively "smooth", meaning they have a structure that can be well captured with frequencies
- The mathematical basis of Fourier is orthogonal, meaning each frequency adds unique information
- We can predetermine how many frequencies we want to keep, thus directly controlling the parameter count

Summary:

Unlike other methods that try to reduce parameter count by limiting matrix rank (like LoRA), this approach looks at the problem from a different angle - through the lens of frequencies, and manages to achieve significantly more compression.

https://arxiv.org/abs/2405.03003

⚡🚀Mike's Daily Paper - 29.11.24: ⚡🚀
In-Context Learning with Long-Context Models: An In-Depth Exploration

The Paper presents comprehensive empirical research on in-context learning (ICL) with language models having long context windows. As a reminder, with ICL, the model receives

several examples demonstrating specific operations and is then asked to perform this operation on new examples.

New findings about ICL behavior in LLMs with long context windows:

1. Continuous performance improvement: Significant performance increase when raising the number of demonstration examples from 10 to 1000 examples
2. Reduced order sensitivity: The impact of example order decreases by 50% with 1000 examples compared to 10 (for random arrangement)
3. Decrease in RAG advantage: The advantage of RAG significantly diminishes with more examples
4. Impact of example clustering by categories: Sorting examples by categories becomes more detrimental to performance as the context window grows
5. Efficiency of short attention lengths: Similar performance can be achieved with a relatively short attention mechanism spanning 50-75 examples
6. Comparison to fine-tuning: In-context learning with long context windows typically matches or exceeds fine-tuning with few examples, but fine-tuning wins when there are enough examples.

https://arxiv.org/abs/2405.00200

🚀Mike's Daily Paper - 30.11.24:

⚡🚀 Fishing for Magikarp: Automatically detecting under-trained tokens in large language models

An interesting paper from Cohere, one of the companies developing foundational language models.

Background: The paper investigates the interesting issue of under-trained tokens - tokens that appear rarely or not at all in the model's training dataset. One possible reason for the existence of such tokens lies in the fact that the tokenizer vocabulary isn't always built based on the dataset the model is trained on.

The tokenizer vocabulary is built on a much smaller dataset than the model's massive pretraining dataset: building a token vocabulary using existing algorithms on datasets of tens of trillions of tokens isn't computationally feasible. Generally, they choose the "most frequent" subwords in the dataset (including punctuation marks etc.) according to a specific method (today the popular method is Byte-Pair Encoding or BPE, another tokenization method is called WordPiece). The differences between the tokenization dataset and the model training dataset can lead to the creation of strange tokens like _TheNitrome.

The presence of under-trained tokens leads to several issues, including wasting tokenizer capacity and reducing model efficiency. Additionally, they can cause unwanted outputs and

disrupt downstream applications, especially in an era where language models increasingly use external data. Such tokens naturally "invite" various jailbreaks. Although some work has been done on identifying these problematic tokens, there's still a lack of reliable and well-explained automated methods tested across a wide range of models.

Research Details: The paper proposes identifying such undertrained tokens through a specific transformation of the unembedding matrix U - the matrix that maps token representations to a vector containing probability distribution for all tokens in the vocabulary.

The authors note that the training loss is minimized when unused tokens are predicted with probability 0, regardless of input, causing their logits to converge to negative infinity. The paper hypothesizes that the model can achieve such input-independent prediction by subtracting a constant vector c from U's rows, leading to a constant negative contribution to the logit values of unused tokens.

[Rest of the technical algorithm details are accurately translated in your previous request]

https://arxiv.org/abs/2405.05417

⚡🚀Mike's Daily Paper - 30.11.24: ⚡🚀
Fishing for Magikarp: Automatically detecting under-trained tokens in large language models

An interesting paper from Cohere, one of the companies developing foundational language models.

Background:

The paper investigates the interesting issue of under-trained tokens - tokens that appear rarely or not at all in the model's training dataset. One possible reason for the existence of such tokens lies in the fact that the tokenizer vocabulary isn't always built based on the dataset the model is trained on.

The tokenizer vocabulary is built on a much smaller dataset than the model's massive pretraining dataset: building a token vocabulary using existing algorithms on datasets of tens of trillions of tokens isn't computationally feasible. Generally, they choose the "most frequent" subwords in the dataset (including punctuation marks etc.) according to a specific method (today the popular method is Byte-Pair Encoding or BPE, another tokenization method is called WordPiece). The differences between the tokenization dataset and the model training dataset can lead to the creation of strange tokens like _TheNitrome.

The presence of under-trained tokens leads to several issues, including wasting tokenizer capacity and reducing model efficiency. Additionally, they can cause unwanted outputs and disrupt downstream applications, especially in an era where language models increasingly use external data. Such tokens naturally "invite" various jailbreaks. Although some work has been

done on identifying these problematic tokens, there's still a lack of reliable and well-explained automated methods tested across a wide range of models.

Research Details:

The paper proposes identifying such undertrained tokens through a specific transformation of the unembedding matrix U - the matrix that maps token representations to a vector containing probability distribution for all tokens in the vocabulary.

The authors note that the training loss is minimized when unused tokens are predicted with probability 0, regardless of input, causing their logits to converge to negative infinity. The paper hypothesizes that the model can achieve such input-independent prediction by subtracting a constant vector c from U's rows, leading to a constant negative contribution to the logit values of unused tokens.

Main Algorithm:

The authors propose the following algorithm for identifying undertrained tokens:

- Define a set S of tokens suspected to be undertrained (i.e., indices of rows in U)
- Calculate the first principal component c of U as an estimate for the constant component c. Since the softmax function is invariant to constant shifts, care must be taken to remove such a constant component to maximize the separation of unused tokens.
- Remove it to get $U' = U - (c^T*U)U$
- Calculate the average embedding vector of unused tokens u_oov = U'_i, $i \in S$. Calculate the cosine distances (or L2 distance) between u_oov and the other rows in U′.

Tokens where this distance is relatively small compared to others (in the 2nd percentile, for example) are suspected to be undertrained tokens. The paper cross-validates the probabilities of tokens suspected to be undertrained and shows that they are very small and change very slowly (mainly due to weight decay) consistently throughout training (regardless of input).

https://arxiv.org/abs/2405.05417

Mike's Daily Paper - 02.12.24:
Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation

History: Today I'll do a small time travel (from my perspective) and review a paper about computer vision. I used to review these more frequently, but lately most papers I review belong to the NLP domain. It's no secret that today, diffusion models (mostly the latent ones) have largely dominated the visual data generation field (images and video).

However, 3-4 years ago, the situation in the visual domain (particularly generative) was quite different. It included VAEs (Variational AutoEncoders), Normalized Flows, but was predominantly dominated by GANs (Generative Adversarial Networks). There were also

interesting combinations of these methods that achieved quite good performance, like VQ-GAN, which combines VAE and GAN.

Background: Today's paper revives VQ-GAN and claims that with slight improvements, it can achieve better results than generative diffusion models of similar sizes (parameter count). This is quite a strong claim that requires understanding what the authors added to VQ-GAN, introduced 4 years ago. First, I'll briefly explain how VQ-GAN works (I reviewed it in detail in Hebrew). Basically, VQ-GAN consists of an encoder that encodes image patches in a latent space, a codebook containing many vectors encoding these patches, and a decoder that converts these patch representations (vectors) back into image patches.

After encoding a patch, the closest vector (using L2 distance I believe) is selected from the codebook and fed to the decoder (along with other patch vectors). The encoder and codebook are trained to return vectors as close as possible to each other (there's also stop-gradient involved to “not to optimize too much stuff simultaneously”), and the decoder is trained to reconstruct the image precisely (for each patch separately and for the whole image) measured by perceptual similarity (LPIPS) and includes a GAN loss computed with the discriminator.

What the Paper Did: But how do we use this for image generation? After completing VQ-GAN training, they take all the latent representations of images from the dataset and train a transformer decoder to predict patch representation given previous patches. This is where our beloved LLMs come in, as the authors train one of the Llamas for this task. After all, we have a dictionary (codebook) like in natural language, just with visual tokens instead of regular ones.

And it works quite well (according to the paper authors)...

https://arxiv.org/abs/2406.06525

Mike’s Daily Paper - 04.12.24:
KAN: Kolmogorov–Arnold Networks

To be honest, it's quite an oversight that in the 7 months since this paper was published, I haven't reviewed it. It already has 400 citations and counting. I personally really love papers based on proven mathematical theorems, and unfortunately, we don't have many of those lately.

This widely discussed paper presents a new architecture based on the Kolmogorov-Arnold theorem, which states that any continuous multivariate function can be represented as a (double) sum of single-variable functions. In simple terms, any function can be represented as a sum of sums of functions, where each function has only one variable.

This theorem is "parallel" to the Universal Approximation Theorems (there are several) which state that any function (meeting not particularly restrictive conditions) can be represented by a neural network with a depth of 2 or more layers. Today's neural networks are built based on UAT (broadly speaking), and the reviewed paper proposes building them based on the KA theorem. Quite naturally, this caught on.

The KAN model is built from layers, each of which is a sum of learned functions (meaning their parameters are learned from the dataset). Each such learned function consists of a linear combination of several b-splines and a parameterless function called silu(x).

A B-spline is a function defined on an interval, divided into several segments (called a grid) which are parameters of the B-spline. It consists of several polynomials (usually degree 3) where each segment has its own polynomial. B-splines are used for function approximation where the coefficients for each segment's polynomial are determined to maximize approximation accuracy. So in KAN, they learn the grid parameters to minimize the problem's loss function.

And that's it - there was quite a bit of excitement around this new architecture, but it turned out that training KAN is not simple at all and doesn't always converge. However, this didn't stop it from getting 400 citations in half a year with dozens of follow-up papers, some of which I'll probably review. Meanwhile, I haven't lost hope in KAN…

Mike's Daily Paper - 05.12.24:
Memory^3: Language Modeling with Explicit Memory

Expanded Analysis of Key Contributions

a. Introduction of Explicit Memory

The paper introduces explicit memory as a quite interesting appendix to the architecture of language models. Unlike the static nature of model parameters or the transient use of working memory (contextual key-value pairs), explicit memory acts as a dynamic, structured knowledge mechanism. This memory type is externalized, meaning the model stores frequently referenced knowledge in retrievable key-value pairs outside its core neural architecture.

Explicit memory addresses a crucial challenge: balancing the size of LLMs with their performance. By externalizing less abstract knowledge (e.g., facts, figures, domain-specific rules) into explicit memory, the model avoids bloating its parameter space while maintaining or even enhancing accuracy. This innovation not only improves computational efficiency but also makes the system modular. Updates to knowledge do not require retraining the entire model, a feature that mirrors human learning processes, where new information is stored without altering underlying cognitive functions.

b. Memory Hierarchy Framework

The **memory hierarchy framework** proposed in the paper is inspired by human cognitive systems, where long-term memory is categorized by accessibility and frequency of use. The authors design this framework to strategically allocate knowledge across three tiers:

1. **Plain Text (High Read Cost, Low Write Cost)**:
    - Suitable for rarely accessed information, plain text storage keeps the overall system lightweight. Retrieval from this tier is less efficient but serves as a fallback

for uncommon queries.

2. **Explicit Memory (Balanced Costs)**:
    - Frequently accessed but not fundamental knowledge is stored in explicit memory, which balances the retrieval speed and storage cost. Its integration with sparse attention ensures that only the most relevant portions of memory are activated, improving inference efficiency.

3. **Model Parameters (Low Read Cost, High Write Cost)**:
    - Reserved for abstract and core knowledge that forms the foundational reasoning abilities of the model. Training updates this layer, making it computationally expensive to modify.

This hierarchy enabled Memory3 to prioritize resource allocation dynamically, ensuring that computational costs remain manageable while delivering high performance. This design is especially relevant for applications requiring on-the-fly knowledge updates, such as customer support systems or domain-specific chatbots.

c. Architecture

The architecture of Memory3 seems a significant evolution of standard Transformer models, integrating **explicit memory** seamlessly. Key innovations include:

1. **Sparse Attention Mechanisms**:
    - By integrating explicit memory into the attention mechanism, Memory3 avoids the quadratic scaling issue associated with self-attention. Sparse attention reduces computation by focusing only on a subset of memory chunks most relevant to the query.
2. **Efficient Memory Retrieval**:
    - The model employs cosine similarity-based search to retrieve relevant key-value pairs. Pre-computed embeddings of memory chunks make retrieval fast and scalable, ensuring that inference speed is not compromised even when the memory size grows.
3. **Memory Sparsification**:
    - To keep memory efficient, the authors introduce techniques like **top-k token selection**, where only the most informative tokens are retained. This is complemented by **vector quantization**, which compresses memory embeddings without significant loss in representational power.

4. **Flexibility in Knowledge Updates**:

- Unlike fixed parameter storage, explicit memory allows modular updates. For example, adding new knowledge involves appending key-value pairs rather than retraining the model, making Memory3 adaptable and future-proof.

d. Training Paradigm

The authors adopt a **2-stage training paradigm** that optimizes the model for explicit memory integration:

1. **Warmup Stage**:
    - During the initial stage, the model undergoes standard pretraining without explicit memory. This phase ensures the development of robust abstraction capabilities and foundational linguistic understanding. The warmup phase mirrors the pretraining of traditional Transformer models.
2. **Continual Training Stage**:
    - In this stage, the model learns to write to and read from explicit memory. Training objectives are expanded to include memory-specific tasks such as:
        - **Memory Writing**: Optimizing the storage of knowledge as key-value pairs.
        - **Memory Retrieval**: Enhancing the ability to retrieve relevant information efficiently and accurately during inference.

The two-stage approach balances the need for general abstraction (via warmup) with task-specific adaptation (via continual training). This ensures that the integration of explicit memory does not disrupt the core language modeling abilities of the system.

Implications and Future Potential

These contributions collectively position **Memory3** as a step forward in building scalable, efficient, and modular AI systems. By incorporating explicit memory, the paper demonstrates a pathway to reduce computational overhead, enhance factual accuracy, and allow flexible updates to knowledge. This architecture could serve as a blueprint for future LLMs, particularly in domains requiring high interpretability and frequent knowledge updates.

Mike's Daily Paper - 07.12.24:
Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering

1. Overview

The paper addresses the limitations of conventional retrieval-augmented generation (RAG) systems by introducing a **knowledge graph (KG)-augmented retrieval system** tailored for

customer service. Authored by a LinkedIn research team, the approach enriches LLMs with structured knowledge from historical customer service tickets. By incorporating both intra- and inter-issue relationships in a graph, the proposed method significantly enhances retrieval accuracy, answer quality, and efficiency, as evidenced by substantial performance gains in metrics like MRR, BLEU, and resolution time in real-world deployment.

2. Key Contributions

a. Integrating KGs into RAG

The authors leverage KGs to overcome critical limitations of traditional text-chunk-based RAG systems:

- **Preservation of Structural Information**:
    - Each ticket is represented as a tree structure (intra-ticket relationships) and linked to others via semantic or explicit relations (inter-ticket relationships).
    - This representation avoids segmentation artifacts, retaining logical coherence across sections of tickets, such as problem descriptions and resolutions.
- **Improved Retrieval and Generation**:
    - By navigating the graph rather than isolated chunks, the system identifies relevant subgraphs to feed into LLMs for answer generation.

b. Knowledge Graph Construction

The KG comprises 2 levels:

1. **Intra-Ticket Tree**:
    - Nodes represent sections like summaries or root causes.
    - Edges capture hierarchical relationships between these sections.
2. **Inter-Ticket Connections**:
    - Explicit connections: Manually annotated relations (e.g., "clone of" or "caused by").
    - Implicit connections: Derived from cosine similarity between embedding vectors of ticket sections.

c. Multi-Step Retrieval and Question Answering

The system operates in 3 phases:

1. **Entity and Intent Extraction**:
    - Parses user queries into entities (e.g., "issue description") and intents (e.g., "steps to reproduce") using LLM prompts and YAML templates.
2. **Subgraph Retrieval**:
    - Computes similarity between query embeddings and KG nodes to identify top-K relevant subgraphs.

    - Dynamically formulates Cypher queries to fetch specific subgraphs for answering.
3. **Answer Synthesis**:
    - Synthesizes responses from the retrieved subgraphs using LLMs, aligning results with user queries.

4. Some related math

The paper introduces a multi-step approach to **customer service question answering** by integrating **KGs** with RAG. It comprises 3 main stages:

1. **Query Entity Identification and Intent Detection**:
    - The system processes user queries by extracting named entities and intents using a predefined graph template and a LLM. Named entities represent key pieces of information (e.g., "issue summary" or "issue description"), while intents capture the purpose of the query (e.g., "fix solution"). For instance, given the query *"How to reproduce the login issue where a user can't log in to LinkedIn?"*, the system identifies the entities as "login issue" and "user can't log in" and the intent as "fix solution."
    - The LLM, guided by structured prompts and the graph template, ensures accurate parsing of queries, accommodating diverse query formulations. This process forms the foundation for precise retrieval and generation in subsequent steps.

2. **Embedding-Based Retrieval**:
    - **Ticket Identification**:
      This step calculates how closely the entities extracted from the user query (e.g., "login issue") match the nodes in the KG. For each entity in the query, the method employs **cosine similarity** to measure the alignment between the entity embedding and the embeddings of nodes in the graph. Scores are aggregated across all nodes belonging to a particular ticket. If a ticket has multiple matching entities, its score increases, making it more likely to be selected as relevant.

    - **Subgraph Extraction**:
      Once the most relevant tickets are identified, these are used to guide database queries written in Cypher, a graph query language. These queries enable the system to extract interconnected subgraphs, such as related descriptions or steps to reproduce an issue. This structured retrieval process ensures the system gathers precise and contextually relevant information from the KG.

3. **Answer Generation**:
    - Synthesizes responses by correlating retrieved graph data with the original query. The LLM acts as a decoder, rephrasing the query dynamically and generating structured answers.

- Example: The query *"csv upload error in updating user email"* is reformulated into Cypher for database interaction, retrieving step-by-step solutions.

The integration of embeddings, dynamic query formulation, and graph-based retrieval ensures precision in identifying relevant data and answering complex customer queries efficiently.

6. Conclusion

This paper pioneers the integration of **KGs** with **RAG** for question answering in customer service. By capturing intra- and inter-ticket relationships, it significantly enhances both retrieval precision and generation quality, setting a new benchmark in practical LLM applications.

https://arxiv.org/abs/2404.17723

Mike's Daily Paper - 09.12.24:
Scaling Synthetic Data Creation with 1,000,000,000 Personas

1. Key Contributions:

Introduction of Persona Hub:

Persona Hub is a repository of 1 billion diverse personas created using scalable techniques.

These personas encapsulate unique knowledge, interests, experiences, and professions, representing ~13% of the world's population.

Persona-driven Synthetic Data Generation:

By integrating personas into prompts, large language models (LLMs) generate highly diverse synthetic data.

Demonstrates applications across varied domains such as math problems, logical reasoning, instructions, knowledge-rich texts, game NPCs, and tool interfaces.

2. Details of Persona Creation: Techniques Used

Text-to-Persona:

Generates personas directly from web data.

Analyzes text context to infer the persona most likely associated with it (e.g., "Who might write or like this text?").

Outputs coarse- or fine-grained persona descriptions (e.g., "a computer scientist" vs. "a machine learning researcher focused on neural architectures").

Scales effortlessly using LLMs and massive public datasets.

Persona-to-Persona:

Expands personas by leveraging relational links (e.g., a child related to a pediatric nurse, or a beggar related to a shelter worker).

Uses relationship-based prompts like "Who is in close association with this persona?"

Enriched personas further by iterating six degrees of separation.

3. Persona Deduplication Process:

MinHash Deduplication: Eliminates similar personas based on text n-gram overlap.

Embedding-based Deduplication: Filters personas using semantic similarity (cosine distance) computed via embeddings. Thresholds for similarity were adjusted depending on quality vs. quantity trade-offs.

After cleaning and deduplication, the repository included 1,015,863,523 unique personas.

4. Applications Demonstrated:

a. Math Problem Synthesis: Created 1.09 million unique math problems using personas.

Fine-tuned a 7B model with these problems, achieving 79.4% accuracy on an in-distribution synthetic test set and 64.9% on MATH, matching GPT-4-turbo-preview.

Demonstrated scalability—adding personas enhanced problem diversity and ensured broad coverage across mathematical concepts.

b. Logical Reasoning Problems:

Synthesized challenging logic puzzles (e.g., spatial or temporal reasoning) tailored to persona characteristics.

Included whimsical Ruozhiba-style problems for testing nuanced logical capabilities.

c. Instruction Generation:

Created user queries reflective of diverse real-world personas (e.g., a chemist might ask for experimental setups; an artist might request painting techniques).

Enabled simulations of multi-turn user-LLM conversations by chaining personas' prompts.

d. Knowledge-rich Texts:

Generated Papers and educational content aligned with personas' expertise (e.g., a horticulturist wrote guides on drought-resistant plants).

Covered almost any topic by leveraging the personas' breadth.

e. Tool (Function) Development:

Predicted tools personas might need (e.g., a cab driver requiring traffic condition APIs).

Generated tool definitions with clear inputs, outputs, and dependencies.

5. Key Results:

Fine-tuned smaller models (e.g., 7B Qwen2) using synthetic data to achieve performance levels typically requiring larger models.

Proved that persona diversity leads to significantly more varied and creative outputs.

Demonstrated that personas could simulate diverse user behaviors, effectively acting as distributed carriers of an LLM's memory.

6. Conclusion:

Persona Hub marks a significant leap in synthetic data generation. Its methodology ensures scalability, diversity, and applicability, creating opportunities for LLM fine-tuning, application development, and even societal simulations.

Mike's Daily Paper - 10.12.24:
LLM2LLM: Boosting LLMs with Novel Iterative Data Enhancement

1. Introduction and Motivation

The paper introduces *LLM2LLM*, a novel framework designed to enhance the performance of large language models (LLMs) in low-data regimes. Fine-tuning LLMs typically demands substantial labeled data, which is often labor-intensive to create. LLM2LLM addresses this challenge with a targeted and iterative data augmentation strategy, leveraging a teacher-student paradigm to dynamically refine training datasets.

Key motivations include:

- Limitations of generic prompting strategies in domain-specific or low-data contexts.
- Challenges in achieving performance gains through conventional data augmentation techniques.
- The need for scalable, cost-effective methods to enhance LLM fine-tuning without significant manual intervention.

2. Methodology

LLM2LLM adopts a teacher-student model framework with three iterative steps:

1. **Student Model Fine-Tuning**: The student LLM is fine-tuned on a small, initial seed dataset.
2. **Error Identification**: The model's performance is evaluated on the training data, and instances where it makes errors are isolated.
3. **Targeted Data Augmentation**: A teacher LLM generates new synthetic examples inspired by the erroneous cases. These examples are then reintegrated into the training dataset for subsequent fine-tuning iterations.

The process emphasizes:

- **Iterative Augmentation**: Refining datasets across multiple cycles rather than generating all augmented data upfront.
- **Targeted Data Focus**: Prioritizing challenging examples that expose the student model's weaknesses.
- **Teacher Model Independence**: The teacher LLM is not required to be superior to the student but must be capable of generating conceptually aligned yet semantically distinct examples.

3. Experimental Results

The framework demonstrates significant improvements across several benchmarks in low-data settings, outperforming traditional fine-tuning and other data augmentation baselines. Results include:

- **GSM8K (Math Reasoning)**: Achieved a 24.2% accuracy improvement over the baseline.
- **CaseHOLD (Legal Reasoning)**: Improved accuracy by 32.6%.
- **SNIPS (Intent Classification)**: Delivered a 32.0% improvement.
- **TREC (Question Classification)**: Boosted performance by 52.6%.
- **SST-2 (Sentiment Analysis)**: Achieved a 39.8% accuracy gain.

The framework's iterative nature proved critical, with ablation studies showing that multiple augmentation steps outperformed single-step augmentation approaches. Furthermore, fine-tuning from scratch in each iteration outperformed continuous fine-tuning, reducing the risk of overfitting.

8. Conclusion

LLM2LLM is a breakthrough framework that redefines data augmentation for fine-tuning LLMs in low-data settings. By iteratively focusing on challenging examples and leveraging teacher-student collaboration, it achieves remarkable performance gains. This method sets a promising direction for enhancing LLM utility in resource-constrained environments.

https://arxiv.org/pdf/2403.15042

Mike's Daily Paper - 18.12.24
Byte Latent Transformer: Patches Scale Better Than Tokens

Intro

I certainly couldn't miss this paper that was published a few days ago and caused quite a stir in the AI community. The paper proposes replacing the static tokenizer present in every language model with a dynamic mechanism that builds new tokens (called patches) - meaning one that builds them depending on context (contextualized).

Rationale:

The rationale here is quite clear - sometimes there are cases where predicting the next few tokens is quite obvious and can be done as a single unit (meaning combining all tokens into one long token or 'patch' as it's called in the paper). And sometimes the situation is reversed and we would want to predict with finer granularity. This is of course impossible in models with a fixed token vocabulary.

Main Idea:

As mentioned, the paper proposes introducing dynamism in building patches (the new tokens). How does it do this? For a given dataset, the paper trains a relatively shallow model at the byte level, where the model's goal is to predict the next byte. Then in their large model, they determine patch boundaries based on byte entropy. Meaning if the byte entropy is either larger than a certain threshold or experienced an increase above a certain threshold compared to the preceding byte's entropy, a new patch is opened. Otherwise, the current patch continues (e.g. byte aggregation into it).

But how does this whole thing work? As I said, the model is byte-level, meaning it's trained to predict the next byte in the text. But instead of looking at the context as an array of tokens, the authors propose replacing it with dynamic patches determined based on entropy as I explained earlier.

In addition to patches, the paper also uses byte representation using n-grams, taking n-grams (n preceding bytes) for a given byte from n=3 to n=8, applying some hash function, summing and normalizing. The result is converted to a vector (the paper doesn't explain how - just mentions there's some linear layer involved) and fed into what's called in the paper Encoder Multi-Headed Cross-Attention (let's call it EMHCA for simplicity).

The purpose of EMHCA is to combine patch representations with their byte representations (each patch only attends its own byte representations, not others'). The initial representation of each patch is computed as pooling (i.e., average) of its byte representations (recall that each patch is an array of bytes). So we're building a representation of each patch that only considers what's inside it (internal representation).

So byte representations and patch representations are fed into EMHCA, which is actually a quite shallow transformer (with few layers) aimed at building a context-dependent representation of patches depending on their bytes. Meaning the byte representations are also keys and values here where the queries are patch representations. As mentioned, what comes out of this shallow transformer is patch representations. Note that EMHCA also outputs byte representations at the end (I couldn't understand how this is built).

All these are input into a deeper and computationally heavier transformer creating a "deeper" representation of the patches. In the final stage, there's the Local Decoder that converts patch representations together with byte representations into final byte representations from which the next byte will be predicted. This is also a shallow transformer but this time patch representations are keys and values and byte representations are the queries.

Performance:

The paper claims various advantages of the proposed method such as the ability to predict more tokens for fixed inference cost, and shows improved accuracy in training models.

Epiloge:

Okay, I have to admit the paper isn't written that well - there are things that weren't explained clearly (imho of course). I just hope I understood it correctly...

https://arxiv.org/abs/2412.09871

Mike's Daily Paper - 19.12.24:
Large Concept Models: Language Modeling in a Sentence Representation Space

Intro:

A second paper (also presented at NeuriPS2024) from Meta proposing a revolutionary concept for language models. While yesterday's paper suggested abandoning the standard tokenizer in language models, today's paper proposes abandoning the next-token prediction we've become so accustomed to in LLMs.

Background:

As you probably remember, LLMs are trained (in pre-training and SFT) by maximizing the likelihood of training dataset D, meaning maximizing the probability of generating D with the trained model. To do this, we maximize (with respect to our language model parameters) the probability of each example in D. Since each piece of data consists of tokens, it can be expressed using Bayes' law as a product of conditional probabilities of each token given the previous tokens (i.e., the context). And that's how we arrive at token prediction given context in both training and inference.

Paper's Main Idea:

The paper emphasizes that we(humans) don't think "token by token" but in concepts when building our speech (while speaking). The paper proposes applying this approach to language models where a concept is defined as a sentence. In other words, the authors propose training a model to predict the next sentence instead of predicting the next token that we're used to in standard language models.

But how do we predict a sentence, given it's discrete and for even a modest sentence length, the number of possible values becomes exponential and too large to perform prediction on (i.e., softmax of enormous size). So the paper suggested performing the next “concept” prediction in a continuous plane and proposes training a model, named Large Concept Model or LCM, to predict the next sentence embeddings given the embeddings of previous sentences in the context window. The paper examines several loss functions, the simplest being L2 between the ground-truth embedding and the predicted one (there are more interesting ones in section 2.4.1 of the paper).

Another way the paper proposed to build the next sentence embedding is training a conditional diffusion model (a very nice idea in my opinion) to predict its embedding.

The embedding is built by an embedder model that remains fixed during training. In addition to the embedder (which is an encoder), there is a decoder that converts the concept (its embedding) to text.

Pretty nice paper, written quite clearly, just a bit too long in my opinion... https://arxiv.org/abs/2412.08821

Mike’s Daily Paper - 20.12.24:
FAN: Fourier Analysis Networks

Brief review of a paper proposing a new architectural layer for neural networks that combines periodic functions like sine and cosine. Periodic functions aren't new in networks - they've been used in Neural Radiance Fields (NeRF) for 3D object and scene modeling.

The paper proposes a layer that linearly combines periodic functions with classical activation functions like sigmoid. The layer is essentially a linear combination of sines and cosines with learnable coefficients alongside standard activation functions. It excels at modeling periodic functions, though its performance on non-periodic functions is less clear (though the paper claims improvements there too).

The paper suggests replacing FFN layers in transformers and gating layers in LSTM with FAN (same weighted sum of sines and cosines with sigmoid) and reports performance improvements across several tasks.

Interesting concept...

https://arxiv.org/abs/2410.02675

Mike's Daily Paper - 20.12.24:
Reasoning in Large Language Models: A Geometric Perspective

This paper investigates the reasoning abilities of LLMs from a geometric perspective, focusing on the connection between the intrinsic dimensionality of input representations and the expressive power of these models. The authors explore how transformer architectures partition input spaces and how this partitioning relates to their reasoning capabilities. By combining theoretical derivations and empirical validations, the work offers valuable insights into how model architecture and context length affect LLM performance on reasoning tasks.

Key Contributions and Detailed Analysis

1. Geometric Framework for Expressive Power

The core idea revolves around the **density of self-attention graphs** and its influence on the **intrinsic dimension (ID)** of inputs to the multi-layer perceptron (MLP) blocks within transformers. The intrinsic dimension, in this context, measures the number of effective degrees of freedom required to represent the input embeddings.

- **Self-Attention as a Graph**:
  The output of the self-attention layer is modeled as a graph, where tokens are nodes and attention values define weighted edges. The density of this graph determines the number of effective connections, which directly influences the intrinsic dimension of the representations passed to downstream MLP blocks.
- **Partitioning the Input Space**:
  Higher intrinsic dimensions enable the MLP layers to partition the input space into finer regions. This allows the model to construct more complex mappings and capture non-linear relationships effectively. As a result, the LLM's reasoning capacity improves with the increased expressive power derived from these finer partitions.
- **Approximation Capabilities**:
  By enabling finer partitioning, higher IDs reduce approximation errors, allowing the MLP to represent complex functions more accurately. This ties directly into reasoning tasks, where nuanced, context-dependent mappings are crucial.

2. Theoretical Insights

The authors rigorously analyze the connection between partitioning, intrinsic dimension, and reasoning capabilities using established mathematical frameworks.

The authors utilize a **continuous piecewise affine formulation** of deep neural networks (DNNs) to explain how input space is partitioned. The main idea of the "Partitioning and Approximation" section is to describe how DNNs break down the input space into multiple regions, each governed by its own specific linear rule.

Neural networks, through their activation functions and layers, divide the input space into several distinct regions. These regions are defined based on how the neurons activate in response to the input data. Think of the input space as a map, and the network creates "zones" or "regions" on this map where each zone has its own unique rule.

Within each region, the neural network behaves like a simple linear function (similar to drawing a straight line). This makes it easier to approximate more complex functions by combining these simpler pieces. The network's ability to approximate complex functions depends on how finely it can partition the input space and define rules for each region. More partitions allow for better approximation, which is critical for tasks like reasoning or decision-making.

This framework helps to understand how neural networks use simple building blocks (linear rules in specific regions) to tackle highly complex problems.

This formulation highlights the ability of DNNs to adaptively partition the input space based on data, with the number of regions correlating directly with the model's approximation power. For transformers, this concept extends to the **self-attention mechanism**, where the density of token interactions influences the induced partitioning of the input space at the MLP level.

Multi-Head Attention Breakdown:
In transformers, multi-head attention splits the attention mechanism into multiple "heads," where each head focuses on a specific aspect of the input. These heads work in parallel to capture different relationships within the data.

Main Theorem:
The theorem shows that the output of a multi-head attention layer can be understood as a combination of regions formed by each individual head. Each head defines a "shape" (technically, a convex hull) based on the transformations it applies to the input.

Main Theorem Terminology: Minkowski Sum
The Minkowski sum is a mathematical operation used to combine these shapes. Intuitively, it means that the overall output of the multi-head attention layer is a space that encompasses all possible combinations of the individual head outputs.

Connection to Intrinsic Dimensions:
This result highlights that adding more heads or making heads more expressive increases the "dimensionality" of the space where the attention outputs lie. This expanded dimensionality enhances the model's ability to represent complex relationships and reasoning processes. The theorem formalizes how the multi-head attention mechanism geometrically partitions and combines input information to enhance the expressiveness and reasoning power of transformer models.

Key Insight:
The effective dimension of this Minkowski sum depends on the density of the attention graph (i.e., the number of active connections between tokens). A higher graph density, achieved through more attention heads or higher connectivity, leads to greater intrinsic dimensionality of

the input to MLP layers. Intrinsic Dimension and Reasoning examines how well a transformer model can capture complex relationships in its input based on the number of meaningful connections it identifies.

Intrinsic Dimension (ID):
The intrinsic dimension measures how many meaningful connections a token in the input sequence has with other tokens. A connection is considered meaningful if its importance (attention weight) exceeds a certain threshold. A higher intrinsic dimension means that more parts of the input are influencing a token. This leads to richer and more detailed representations of the input, enabling the model to better understand complex patterns and relationships. When the model has a high intrinsic dimension, it can effectively "partition" the input space into more regions, allowing it to capture finer details and nuances. This is crucial for reasoning tasks, where understanding subtle relationships is key.

Practical Implications:
Increasing the number of attention heads or providing longer input contexts can raise the intrinsic dimension. This enhances the model's reasoning capabilities without requiring changes to its architecture or training process. The intrinsic dimension reflects how deeply a transformer engages with its input. The richer the connections, the better the model can reason and perform complex tasks.

https://arxiv.org/abs/2407.02678

Mike's Daily Paper - 23.12.24:

T-FREE: Tokenizer-Free Generative LLMs via Sparse Representations for Memory-Efficient Embeddings

We're returning to the topic of tokenizers - it turns out it's hotter than I thought. I came across an interesting paper presenting another tokenization method based on n-gram hash functions. The proposed method aims to address the massive size of vocabularies accompanying large language models (tens of thousands of tokens at minimum) and also the issue of very similar tokens in terms of letters requiring different embeddings, which is inefficient (according to the authors).

The authors introduce a tokenization method where the encoding consists of the following steps:

- Breaking down text into what they call tokens, where in T-FREE these tokens are actually words
- Each word is split into a series of overlapping 3-grams. For example, the word "hello" is represented by five 3-grams: {_*He, Hel, ell, llo, lo_*}. The number of 3-grams in this representation is usually equal to the number of letters in the word

- Each 3-gram is encoded with m hash functions, each of which can take v possible values, where v is one of the method's hyperparameters. Thus, each word is encoded by n*m numbers between 0 and v, where n is the word length (number of letters). The word representation is the average (and rounding) of all these nm values.
- Each value between 0 and v is encoded by a learned vector, where these v vectors essentially constitute the method's vocabulary.

The training and decoding stages (i.e., generating words) appear more complex. First, during training, the goal is to predict the nm hashes of the 3-grams of the next word. So instead of a multi-class problem in regular tokenization decoding (predicting a single token from the token vocabulary), we have here a multi-label problem where we predict n*m hash values. Note that n depends on the word length, meaning we have a different number of "labels" according to word length.

The decoding isn't really clear to me, honestly. When we want to predict the next word, we first compute all the hashes for all possible words (that's quite a lot because each word has all its inflections at least, and additionally, words of different lengths are encoded with different numbers of n*m hashes). Then we choose the word represented by the hashes with the "highest probability." Remember that the model predicts the probability of each hash value from 1 to v (vocabulary size), and it's not entirely clear how the set of hashes with the highest probability is chosen.

In short, nice paper but I'm not clear about the decoding part...

https://arxiv.org/abs/2406.19223

Mike's Daily Paper - 25.12.24:
Vision Language Models are Blind

A nice paper arguing that visual language models are quite blind - in other words, they have no chance of passing an exam at a licensed optometrist. Here are some facts about their failing tests:

1. Visual Language Models (VLMs) cannot reliably determine whether two lines (or two circles) intersect, especially when they're close together. The accuracy in identifying 0, 1, or 2 intersection points between two 2-segment piecewise linear functions ranges from 47% to 85%. In the same two-circle task, the models perform better (accuracy of 73-93%) but still far from the expected 100%.

2. VLMs can perfectly identify a circle and a word separately, but when the word is inside the circle, the models tend to struggle identifying which letter is circled.

3. Vision-language models can accurately count shapes, for example, circles, squares when they are separate and far apart. However, all models struggle to count intersecting circles (like the Olympic logo), and generally, basic shapes that are overlapping or nested.

4. When arranging squares in a grid pattern, we discover that VLMs surprisingly fail to count the number of rows or columns in the grid, whether empty or containing text. This is surprising considering that the models perform so well (accuracy ≥ 90%) on the DocVQA dataset which includes many questions with tables (probably overfitting).

5. When the model is asked to trace colored paths in a subway map with up to 8 routes and a total of 4 stations, VLMs often fail to identify where a path ends, demonstrating accuracy of only 23% to 50%.

6. The GPT-4o model outperforms Gemini-1.5 Pro and Claude-3 Sonnet in 7 complex VLM benchmarks but performs significantly worse in the tasks examined in this paper, where Gemini-1.5 Pro and Sonnet-3.5 are the best performers. In other words, the paper reveals surprising limitations of vision-language models that weren't measured in regular benchmarks.

Frankly, it seems that these VLMs need glasses…

https://arxiv.org/abs/2407.06581

Mike's Daily Paper Review - 27.12.24:
RL for Consistency Models: Faster Reward Guided Text-to-Image Generation

Introduction:

It's been a while since I reviewed papers on diffusion models, so when I came across this interesting paper combining generative diffusion models with Reinforcement Learning (RL), I knew it would be today's review. The paper developed a training method for a generative diffusion model type called a Consistency Model (CM).

Diffusion Models Training with Reinforcement Learning
First, let's address why we need to train generative diffusion models using RL techniques. We already have standard methods for training diffusion models that have achieved impressive performance (in text-to-image generation). You probably know that training diffusion models for image generation is expensive and time-consuming. Using RL for training (or fine-tuning) diffusion models can save us time when we need to train a specialized diffusion model (for example, for a niche domain).

One example of such a task is training a model to create images from prompts (textual descriptions) when we have a function that evaluates how well the image matches the prompt. You can probably guess that this function will serve as our reward function.

Consistency Diffusion Models:
I mentioned that the paper combines a (relatively) new method for training diffusion models called CM, invented by Ilya Sutskever et al., which enables faster generation with generative diffusion models. In very broad terms, this method tries to train a model that enforces

consistency between images reconstructed by the model from noisy images with different noise levels. In other words, they take an image, add noise (usually Gaussian) with different variances, and train a model to return the same clean image (consistency for its own sake).

Why does this method enable faster image generation? Because it essentially allows generating a clean image from noise in just one iteration (that's how the model is trained). In practice, this is done in several iterations (a small number). You start with noise, generate an image from it, add less noise to the generated image, generate from the noisy image again, and continue like this for a few iterations (tens). This speeds up the generation process because standard diffusion models typically need hundreds of iterations.

The proposed method:
Okay, after this long introduction, let's describe what they did in the paper. The authors defined a Markov Decision Process (MDP) describing the image generation process (or any other data actually). As mentioned, a reward function is given to us and measures how well the generated image matches the prompt. The paper defines:

- State s_t as a triplet of the image generated at iteration t, noise strength, and prompt c
- Action a_t is the image at iteration t + 1
- The policy is a conditional probability of an image from iteration t+1 given the generated image from iteration t plus noise
- The initial state is standard Gaussian noise and the reward function is given to us

After defining the MDP for the image generation process, we can use Direct Preference Optimization (DPO) to train a consistency function (= the model we're training). DPO trains a model that maximizes the reward function while constraining the size of model parameter updates in each iteration (invented by Joe Schulman, former CTO of OpenAI).

The paper also claims that such training is computationally efficient and data-efficient (meaning it can work with small datasets).

https://arxiv.org/abs/2404.03673

Mike's Daily Paper Review - 27.12.24:
Position: Future Directions in the Theory of Graph Machine Learning

This position paper argues that while Graph Neural Networks(GNNs) have seen significant success in practice, our theoretical understanding of them remains incomplete and somewhat disconnected from practical applications. The authors identify three key areas that need deeper theoretical investigation:

1. **Expressiveness -** What patterns and structures can GNNs actually represent?
2. **Generalization -** How well do GNNs apply their learning to new, unseen graphs?

3. **Optimization** - How do training dynamics affect GNN performance?

KEY POINTS ON EXPRESSIVENESS

Current Limitations:

- Most theoretical work focuses on binary questions (can a GNN tell if two graphs are different?) rather than quantitative measures (how different are two graphs?)
- Analyses are often limited to specific GNN architectures and don't consider practical variations
- Results don't account for continuous node/edge features that are common in real applications

Proposed Directions:

- Develop metrics to measure similarity between graphs that align with how GNNs process them
- Study how architectural choices (like activation functions and normalization) affect expressiveness
- Create uniform results that work across different graph sizes
- Focus on practically relevant graph types (like molecular graphs)

GENERALIZATION INSIGHTS

Current State:

- Existing theoretical bounds are often too loose to be practical
- Analysis typically ignores graph structure and optimization process
- Results don't explain why more complex GNNs sometimes generalize better

Needed Research:

- Understanding how graph structure affects generalization
- Analyzing out-of-distribution performance (especially on larger graphs)
- Developing better data augmentation techniques for graphs
- Studying how architectural choices influence generalization

OPTIMIZATION CHALLENGES

Key Issues:

- Limited understanding of how gradient descent works for GNNs
- Unclear why certain architectural choices (like normalization) help or hurt
- Sometimes random GNN parameters work as well as trained ones

Research Directions:

- Study convergence properties with realistic activation functions
- Understand how graph structure affects optimization
- Investigate why deeper GNNs are harder to train
- Analyze the role of normalization techniques

PRACTICAL IMPLICATIONS

The authors emphasize that theoretical advances should connect to practical needs:

- Development of standardized benchmarks and evaluation protocols
- Creation of efficient implementations of theoretically-motivated architectures
- Better understanding of domain-specific requirements
- Integration with emerging technologies like large language models

WHY THIS PAPER MATTERS

This paper is significant because it:

1. Identifies crucial gaps between theory and practice in GNN research
2. Provides a roadmap for future theoretical work that could improve practical applications
3. Emphasizes the need to consider all three aspects (expressiveness, generalization, optimization) together
4. Advocates for making theoretical advances more accessible to practitioners

For readers with basic GNN knowledge, this paper highlights why theoretical understanding matters and how better theory could lead to more effective practical applications. While some technical details might be complex, the core message about needing more practical and comprehensive theoretical frameworks is clear and important.

https://arxiv.org/abs/2402.02287

Mike's Daily Paper 12/30/24
Graph Diffusion Policy Optimization

3 days ago I reviewed a paper about diffusion models trained using Reinforcement Learning (RL) methods. 2 days ago I reviewed a paper about Graph Neural Networks(GNNs), and today I decided to review a paper that combines these 3 things (almost). The paper reviewed today proposes a method for training a graph-generating model using diffusion models trained with RL methods (true, there's no pure GNN here but at least we have graphs...)

First, we need to understand how to leverage diffusion models for graph generation. Actually, it's quite intuitive and similar to image generation. You remember that diffusion models are trained to generate an image from pure noise (usually) by gradually reducing its noisy component until it becomes a data piece distributed according to the training dataset distribution(hopefully). This is

a very general explanation, and there are newer approaches that do it differently, like Consistency Models that we discussed in one of the previous reviews.

Can we do something similar with graphs? Turns out we can. We can start by sampling a random graph (meaning sampling its vertices and edges) and train a model to change the values in vertices and edges so that the graph becomes "similar" to the graphs from the train set and also gets a high value according to some reward function (remember, the paper is also about RL, so we have a reward function as well). BTW, there's an implicit assumption in the paper that a vertex can take a finite number of values (say from 0 to a) and each edge can be of several types (i.e., from 0 to b). This means the distributions we're sampling from are categorical, which is different from what we're used to seeing in generative diffusion models for images.

Of course, several questions immediately arise regarding this process:

- How do you randomly sample a graph during inference? This is a very old topic extensively researched by mathematicians, the paper doesn't elaborate on that too much. By the way, during training we sample a graph from the dataset and “add noise” to it by making "random changes" to vertex values and edge types
- How do you compare graphs, meaning how do you understand that a graph received during generation is similar to a graph from the dataset? There are many approaches to compare graphs by comparing their subgraphs or comparing their Laplacians for example
- Choosing a reward function in the graph domain isn't trivial at all. For instance, for tasks of generating graphs for new molecules, one measure of the generated graph's quality is its novelty relative to existing graphs, its effectiveness in treating certain diseases, or its synthetic accessibility (how easy it is to synthesise it). The reward can also be chosen as a similarity function to existing graphs.

Okay, so we have a function C for comparing graphs and a reward function for evaluating graph quality - how do we train a diffusion model? Actually, quite similar to what I described in my review from 3 days ago of the paper: "RL for Consistency Models: Faster Reward Guided Text-to-Image Generation."

First, we need to define the Markov Decision Process(MDP) for training a diffusion model on graphs. And it turns out it's very similar to the paper I mentioned:

- The state s_t as a pair of a generated graph at iteration T-t and also the value T-t, the action a_t is the graph at iteration T-t-1
- The policy (probability of a_t given s_t) is the conditional probability of a graph from iteration T-t-1 given a graph at iteration T-t
- The initial state is a random graph at iteration T and reward function r that is given to us calculated on the final graph at iteration 0

The paper proposes two methods for training a diffusion model for graph generation: The first is classic REINFORCE algorithm which is essentially a policy gradient method that trains a policy to have high rewards. In practice, we sample K iterations between 1 and T and maximize an average product (over K samples) of the policy function (conditional probability of a graph from iteration T-t-1 given a graph at iteration T-t) and the reward r for the generated graph (at iteration 0).

The second proposed method is Policy Optimization where instead of maximizing the policy in its pure form, maximizes the probability of generating graph G_0 from the dataset (which is sampled from the dataset, noised and then the model learns to remove the noise from it) multiplied by the reward for the generated graph. Here too there is averaging over K iterations from which we build an estimation of G_0.

That's it - a bit of a heavy mathy review, hope you managed to understand something from it...

https://arxiv.org/abs/2402.16302

Mike's Daily Paper - 01.01.25:
Inference-Aware Fine-Tuning for Best-of-N Sampling in Large Language Models

We're starting the new year with a review of a quite interesting paper that proposes a method for improving language model training. The major advantage of this method is that it allows adapting the training to the inference approach, and it's quite clear that if done successfully, this should yield better inference quality. In other words, if we use a certain approach during inference - either choosing the "best answer" from among N model responses (the paper develops methods only for this approach and calls it BoN) or self-correction - then we should adapt the training accordingly.

First, the paper formulates two inference-aware (IA for short) objective functions for training, one for SFT and one for RLHF, named IA-SFT and IA-RL respectively. For IA-SFT, we maximize the likelihood of expert answers to questions from our dataset given an inference policy I (which is BoN actually). Essentially, we optimize the policy (which is the LLM's prediction mechanism, or simply the LLM itself) to perform BoN in the best possible way on our dataset. For IA-RL, the goal is to optimize the policy (the LLM) under inference technique I (BoN) to maximize a reward function R.

The paper then precisely defines what BoN is (formula 1), where the goal is to maximize the model's answer quality when we choose the answer according to what's called a verifier score r (= score for answer quality). By the way, there are two additional parameters here: the number of answers N from which we choose the best answer, and the temperature T of the language model. Intuitively, as T increases (more randomness and creativity in answers), the number of answers N should increase.

Here we have the classic trade-off between exploration and exploitation. The higher T is, the more exploration we perform (more diverse answers), while lower T allows us to "enjoy" what we've learned so far (choice of N affects the trade-off inversely to T).

Okay, but in the IA-SFT optimization problem, we still have argmax (used to select the best answer), which makes it very difficult to solve. Despite having argmax estimation methods like softmax and Gumbel softmax, these methods are still inaccurate and computationally heavy (according to the paper). So the authors use a widely used ML trick - approximating the objective function with a variational approximation that turns it (its log) into a sum of the policy (probability of answer y given question x with the model) and a regularization term called inference-aware term. This term is actually the win-rate of answer y on question x over the current model, where the score of each pair (x, y) is computed with the verifier score function r (with a normalization constant).

In IA-RL, things get a bit more complicated, and the authors use a result from one of Joe Schulman's papers (former CTO of OpenAI) with the legendary Sergey Levine and Peter Abbeel to get a gradient estimate that takes a form similar to the old and well-known REINFORCE algorithm, meaning the product of the policy log with a reward function ("centered" with the expectation of reward function to reduce variance). The paper also discusses interesting cases of optimizing the IA-RL objective function for several forms of verifier score r (for example, binary).

A mathematically heavy paper, I tried (to the best of my ability) to make it a bit more accessible for you...

https://arxiv.org/abs/2412.15287

Mike's Daily Paper - 02.01.25:
Loss of plasticity in deep continual learning

Today we're briefly reviewing a fairly light paper from Nature.

Introduction: Standard deep learning methods show a gradual decline in their ability to continuously learn new tasks (gradually "adding" tasks to the model). Unlike catastrophic forgetting, where previous knowledge is lost, loss of plasticity limits the network's ability to efficiently learn new tasks. Comprehensive experiments on datasets like ImageNet and CIFAR-100, as well as Reinforcement Learning scenarios, revealed that neurons become dormant (unchanging across all examples) or overly specialized in specific tasks, reducing their ability to adapt to new data. Over time, networks experiencing continual learning perform no better than shallow (linear) models, highlighting a fundamental limitation of gradient descent-based methods for continual learning (and we train models with GD today)....

Gradient Descent for Continual Learning: Continual learning methods try to address loss of plasticity by reinitializing dormant neurons (those that "almost never fire") and retraining them

using gradient descent. This approach attempts to "create" neurons that will learn new tasks without getting locked into specific tasks, allowing it to learn new tasks without significant performance degradation. Unlike conventional methods relying solely on gradient descent, GD for continual learning is characterized by gradual updates to different sets of model weights, similar to what happens in biological learning systems.

Additional Training Methods: As mentioned, loss of plasticity is related to over-optimization (according to the paper) of weights and the emergence of dormant neurons in the network. These neurons either stop contributing to learning (for ReLU activation) or enter saturation (reaching 0 or 1 for sigmoid). Techniques like L2 regularization reduce model weight growth and maintain "plasticity" (flexibility for new tasks) to some extent. For example, the Shrink and Perturb method, which combines regularization with small random changes in weights, reduces the phenomenon of dormant neurons and thus increases the model's learning capacity.

Continual Learning Challenges in RL: Continual learning is essential for RL even more than in supervised learning. Not only can the environment change, but the behavior of the learning agent can also change, thereby influencing the data it receives even if the environment remains stationary. For this reason, the need for continual learning is often more apparent in reinforcement learning, and reinforcement learning is an important setting for demonstrating deep learning's tendency toward loss of plasticity. The paper examines using the methods we discussed earlier for RL tasks along with PPO, the famous optimization algorithm in RL.

https://doi.org/10.1038/s41586-024-07711-7

Mike’s Daily Paper - 03.01.25:
A PERCOLATION MODEL OF EMERGENCE: ANALYZING TRANSFORMERS TRAINED ON A FORMAL LANGUAGE

Introduction:

Modern neural networks, particularly LLMs, exhibit a wide range of capabilities, which allow them to serve as foundational systems for various downstream applications. This paper proposes a phenomenological definition of emergence within the context of neural networks, focusing on how specific structures underlying the data-generating process can lead to sudden performance improvements for narrower tasks.

Emergence in Neural Networks

The authors define emergence in deep neural networks as the acquisition of specific structures that cause sudden performance growth on specific tasks. They empirically investigate this through an experimental system grounded in a context-sensitive formal language, demonstrating that transformer-powered models trained on strings from this language display emergent capabilities. Once the model learns the underlying grammar and structures, performance on related tasks improves significantly.

Formal Language Definition

The experimental system utilizes a probabilistic context-free grammar (PCFG) to define a context-sensitive formal language. The grammar consists of:

Terminal Symbols: Parts of speech including subjects, objects, verbs, adjectives, adverbs, conjunctions, and prepositions.

Non-terminal Symbols: Symbols that define the structure of sentences.

Production Rules: Rules that dictate how terminals and non-terminals can be combined to form valid sentences.

The model is trained on tasks such as free generation, unscrambling(fixing wrong text), and conditional generation, with performance metrics tracked throughout the training process.

Tasks and Evaluation Protocols

1. Free Generation: The model generates sentences that adhere to the grammatical rules.

2. Unscrambling: The model reorders a scrambled string of words to form valid sentences.

3. Conditional Generation: The model creates sentences based on given entities or properties.

Evaluation metrics include grammaticality checks, type checks, exact match accuracy, per-token accuracy, and more, providing a comprehensive assessment of the model's capabilities.

Learning Dynamics:

The results reveal three distinct phases in the learning dynamics of the model:

1. Initial Phase: The model learns basic grammatical structures with minimal performance improvement.

2. Phase Change: A sudden increase in performance occurs once the model begins to understand the relative type constraints.

3. Generalization Phase: The model demonstrates improved performance on tasks, indicating a transition from memorization to generalization.

Emergent Capabilities:

The authors observe that as the language model learns the grammar and type constraints, significant performance boosts are observed across various tasks, particularly in unscrambling and conditional generation. The presence of specific structures allows the model to infer valid combinations of entities and properties, leading to emergent capabilities.

Scaling and Transition Points:

The paper discusses how the point of emergence scales with the number of properties in the language. The transition point, where significant performance improvements occur, is linked to the scaling of descriptive properties, allowing for predictions about when capabilities will emerge as the model continues to learn.

Conclusion:

This research contributes to the understanding of emergence in neural networks by establishing a framework that defines and characterizes emergent properties based on the acquisition of underlying structures. The findings indicate that grammatical and type constraints serve as critical factors in predicting the emergence of capabilities in language models.

Mike's Daily Paper - 06.01.25:
A Survey on Efficient Inference for Large Language Models

The paper provides a comprehensive overview of methods for optimizing the efficiency of LLMs during inference. :

Key Challenges Addressed:

1. Model Size: LLMs, often exceeding billions of parameters, have substantial computational and memory demands.
2. Quadratic Complexity in Attention: The self-attention mechanism's quadratic complexity in input sequence length significantly impacts inference latency and memory usage.
3. **Auto-Regressive Decoding:** Token-by-token generation exacerbates memory access inefficiencies and limits effective throughput.

Taxonomy of Optimization Techniques:

The paper classifies methods into three levels:

1. Data-Level Optimization:

Input Compression: Techniques like prompt pruning, prompt summarization, soft prompt-based compression, and retrieval-augmented generation reduce the size of input prompts while retaining semantic information. This is particularly effective for scenarios requiring longer inputs.

Output Organization: Methods such as Skeleton-of-Thought (SoT) and dependency graph-based approaches enable partial parallelization of token generation, leveraging the inherent structure of LLM outputs.

2. Model-Level Optimization:

Efficient Structure Design:

Mixture-of-Experts (MoE) models dynamically allocate computational resources to input tokens, optimizing Feed-Forward Networks (FFNs).

Simplified or kernel-based attention mechanisms reduce computational complexity from the quadratic one.

Alternatives to transformers, such as State Space Models (SSMs) and recurrent architectures, address inefficiencies while maintaining competitive performance.

Model Compression:

Quantization: Reduces bit-width for weights and activations. Post-training and quantization-aware training methods preserve accuracy despite compression.
Sparsification: Removes redundant parameters or attention heads, using techniques like pruning or sparse attention.
Knowledge Distillation: Trains smaller models to mimic the behavior of larger ones, with minimal performance loss.

3. System-Level Optimization:

Improvements in inference engines (e.g., speculative decoding and offloading strategies) and serving systems (e.g., batching, scheduling, and memory management) enhance hardware utilization and throughput.

Efficiency Analysis:

The inference process is divided into two stages:

1. **Prefilling:** Initialization of the model with input prompts and caching key-value pairs for subsequent steps.
2. Decoding: Sequential token generation with memory and computational overhead.

Efficiency metrics such as latency (per token and total sequence), memory usage (model weights, KV cache, peak memory), and throughput (tokens/second, requests/second) are analyzed to quantify the impact of optimization methods.

Experimental Insights:

-The survey consolidates experimental results across methods to provide comparative insights into trade-offs between efficiency gains and accuracy. For instance:

Quantization techniques achieve substantial memory savings but may require careful calibration to avoid accuracy degradation.Structured sparse attention mechanisms balance computational efficiency and model fidelity, especially for long-context tasks.

Future Directions:

1. Adaptive techniques that dynamically adjust model size and computation based on input complexity.
2. Co-optimization across the data, model, and system levels to maximize efficiency.
3. Hardware-aware methods to exploit modern accelerators like GPUs and TPUs.

https://arxiv.org/abs/2404.14294

Mike's Daily Paper - 07.01.25
Anchored Preference Optimization and Contrastive Revisions Addressing Underspecification in Alignment

Introduction:

Today's paper discusses an improvement to an alignment method for language models called DPO, which belongs to the RLHF (Reinforcement Learning with Human Feedback) family of techniques. As you remember, RLHF is one of the stages (usually the last) in training LLMs, along with pretraining and Supervised Fine Tuning (SFT).

RLHF Fundamentals:

RLHF's goal is to show the model the difference between preferred (by humans) and less preferred responses. In more mathematical terms, RLHF trains the model to maximize the ratio between the score of the more preferred (better) response to a less preferred one. Classical RLHF method Proximal Policy Optimization adds a regularization term to the maximization objective that tries to keep the learned policy (like a trained LLM) close to the initial LLM (proximity is calculated using KL divergence on the distribution of tokens predicted by both models).

The score is computed using a reward model trained (in the stage prior to RLHF) to estimate the "quality" of the response to a given question. In other words, a reward model R should give a high score to a good response and a low score to a less good response. The model is trained on pairs of good and not-so-good responses to questions, where typically the responses are labeled by human annotators (sometimes powerful language models are used for this labeling).

DPO Methods:

It turned out that the PPO optimization objective could be approximated without training a reward model. In the last two years, several papers have proposed methods that can manage without a reward model. One of them is DPO, which stands for Direct Preference Optimization. With DPO, the reward function r_dpo is defined as the logarithm of the ratio between the policy (the predicted token distribution measured by the model or likelihoods) for the optimized model (being fine-tuned) to that of the initial model. DPO's training objective is to maximize the expectation (over the dataset of question-answer pairs) of the difference of r_dpo between preferred and less preferred responses.

Main Paper Ideas:

The main point of the paper is the observation that optimizing DPO's objective function can affect the ratio of likelihoods of preferred responses w to less preferred ones l in different ways. It can increase the difference between them (which is its stated goal) but it can increase p_w more than it increases p_l, or decrease p_l more than it decreases r_w. These scenarios might lead to very different models. The paper notes that a preferred response isn't necessarily better than what the model produces before alignment. In this case, DPO might harm the model's performance.

The paper examines the different cases of r_dpo values for responses w and l (preferred and less preferred respectively) and constructs two objective functions for DPO that might lead to better performance for these cases. The training method optimizing these functions was named Anchored Preference Optimization or APO. The first proposed function increases the policy value (response likelihood) when the current value of r_dpo for w is close to 0 (w has lower likelihood for the initial model) and further decreases the likelihood of the less preferred response if r_dpo for l is close to 0.

The second proposed function, on the other hand, decreases w's likelihood when r_dpo is close to 0 for w and increases the difference between the likelihoods of w and l when the difference between r_dpo for w and l is close to 0. All this is aimed at making the language model trained using DPO converge to a better solution.

There's something else interesting in this paper. The authors claim that for DPO to work better, both responses (w and l) need to be relevant to the question and one should be "just a little" better than the other. In other words, like in contrastive learning, it's better to train the model on hard negatives.

The authors propose a method for identifying (and building a dataset of) preferred and less preferred responses by creating a preferred response from any relevant response by using an LLM that improves the response (with an appropriate prompt). Another method the authors suggest is to use two responses from the trained model (with DPO) and apply a language model aimed at determining which response is better (called an on-policy judge). One can also build a dataset offline with a third language model and a judge model.

Long review - I hope you survived...

https://arxiv.org/abs/2408.06266

Mike's Daily Paper - 09.01.25
When Can Transformers Count to n?

The paper explores the theoretical and empirical limits of transformer architectures when applied to simple counting tasks. Specifically, it examines tasks like "Query Count" (QC) and "Most Frequent Element" (MFE) to determine when transformers can efficiently solve these

problems. The study reveals both the capabilities and inherent limitations of transformers in such contexts, providing critical insights into their architectural constraints.

Summary of Key Contributions:

QC Task
The QC task involves determining how many times a specific token appears in a sequence. The authors demonstrate that transformers can implement this task efficiently if the embedding size d exceeds twice the vocabulary size m:

- For d > 2m, a histogram-based approach enables counting by embedding tokens orthogonally. This allows the model to construct a histogram of token occurrences in a single transformer layer.
- For d < m, the orthogonality of embeddings breaks down, making exact counting infeasible. The paper rigorously proves this limitation using the Welch bounds, which quantify the trade-offs in embedding dimensionality.

When the embedding size d is smaller than the vocabulary size m, it becomes impossible to construct orthogonal embeddings for all tokens. To address this, the authors propose the CountAttend Solution, which uses attention mechanisms to estimate token counts. However, this solution introduces specific requirements and practical challenges.

CountAttend Solution
When d < m, the authors propose the "CountAttend" solution, which leverages attention mechanisms to count tokens:

The CountAttend solution involves two main components:

1. **Attention Weights**

- The attention mechanism generates weights that encode the relationship between the query token and all tokens in the sequence.
- For counting, these attention weights must be inversely proportional to the count of the token in the sequence.For example, if a token appears c times, the attention weight associated with it should be proportional to 1/c. This weighting ensures that each token's contribution to the output is normalized by its frequency.

2. **MLP to Invert Weights**

- To recover the actual count c from the attention weights, an MLP (Multi-Layer Perceptron) is required. The MLP essentially performs the inverse operation, mapping the attention weights back to the counts. This requires the MLP to learn a function of the form: f(w) = 1/w where w is the attention weight for a token.

Challenges with the CountAttend Solution

1. **Attention Weight Calculation:**
    - Computing attention weights inversely proportional to token counts requires precise modeling of token relationships across the sequence. This adds complexity to the attention mechanism.
2. **MLP Size:**
    - For longer sequences, the number of neurons in the MLP must grow proportionally. This scaling is computationally expensive and memory-intensive, making it impractical for sequences with many unique tokens or high variability in token frequencies.
3. **Generalization Issues:**
    - The MLP must learn a continuous mapping that accurately recovers counts for all possible token frequencies. This is challenging, especially for tokens with high or very low frequencies.

MFE Task

For the MFE task, which involves finding the token with the highest frequency, the authors show that the task can be implemented if d=O(m using a histogram-based approach. For d < m, the task becomes infeasible, as shown through communication complexity arguments. The authors also propose a two-layer transformer solution, which eliminates the need for a large MLP but increases the model's depth.

Phase Transition in Performance

The paper identifies a critical phase transition: transformers fail at counting tasks when d < m. This threshold underscores the trade-offs between embedding size, vocabulary size, and task complexity..

Theoretical Insights

1. **Orthogonal Embedding Construction**
   The authors leverage the mathematical properties of orthonormal embeddings to implement histogram-based counting. For d > m, embeddings can be constructed such that the dot product between distinct token embeddings is zero. This ensures exact token counting within a single attention layer.
2. **Welch Bounds**
   The paper uses Welch bounds to show that, for d < m, the inner product between embedding vectors becomes non-negligible, introducing errors in the histogram. This result formalizes the failure of histogram-based solutions in low-dimensional embeddings.
3. **Communication Complexity**
   For the MFE task, the authors employ a communication complexity framework to prove that transformers require d=Ω(m) to solve the task. This lower bound establishes that embedding size must scale with vocabulary size to maintain performance.

Practical Implications

The findings have several implications for the design and deployment of transformers in real-world applications:

1. **Architectural Scaling**
   Transformers must scale embedding sizes with vocabulary size to perform counting tasks effectively. For tasks involving large vocabularies or long sequences, alternative architectures or hybrid approaches may be necessary.
2. **Positional Embeddings**
   The authors emphasize the necessity of positional embeddings for counting tasks. Without positional information, transformers cannot distinguish between sequences of varying lengths, leading to invariance issues.
3. **Efficiency Considerations**
   While the histogram solution is efficient for d>m, its practical implementation may be constrained by memory and compute limitations, particularly for very large vocabularies.

Conclusion

The paper provides a comprehensive analysis of transformers' capabilities and limitations in solving basic counting tasks. By combining rigorous theoretical proofs with empirical validation, it highlights the architectural trade-offs inherent in transformer models. The work underscores the importance of embedding dimensionality, positional information, and model depth in enabling transformers to handle fundamental tasks.

Future research:

- Hybrid architectures that combine transformers with symbolic methods for counting tasks.
- Extensions to tasks involving hierarchical or structured counting.
- Mechanistic interpretability studies to elucidate the internal representations learned by transformers during counting tasks.

https://arxiv.org/pdf/2407.15160

Mike's Daily Paper -10.01.25
Chain of Thought Empowers Transformers to Solve Inherently Serial Problems

The paper presents a theoretical analysis of how Chain of Thought (CoT) prompting enhances transformer models' ability to handle serial computation). The authors make significant contributions by establishing formal expressiveness bounds and introducing a novel complexity class that characterizes transformers' computational capabilities with CoT.

The paper's primary theoretical contribution lies in its expressiveness bounds. Through rigorous mathematical analysis, the authors demonstrate that constant-depth transformers with constant-bit precision are limited to solving problems in AC0(consists of all families of circuits of depth O(1) and polynomial size, with unlimited-fanin AND gates and OR gates) without CoT (Theorem 3.1). However, they show that with T steps of CoT, transformers become capable of

solving any problem computable by boolean circuits of size T (Theorem 3.3). This result has profound implications, as it establishes that polynomially many CoT steps enable transformers to compute any function in P/poly. Furthermore, the authors prove that transformers with linearly many CoT steps can compute all regular languages, including challenging problems like the composition of non-solvable groups such as S5.

The theoretical framework developed in the paper rests on three pillars. First, the authors present a comprehensive finite precision analysis, introducing a rigorous treatment of floating-point computation in transformers. This includes careful handling of rounding issues through IEEE 754-style arithmetic and the development of correct rounding operations that respect computational reality. Second, they establish deep connections with complexity theory by defining a new complexity class CoT[T(n), d(n), s(n), e(n)], which characterizes transformer computation based on the number of CoT steps T(n), embedding dimension d(n) , precision bits s(n), and exponent bits e(n). Third, they integrate automata theory by utilizing the Krohn–Rhodes theory to analyze transformer capabilities and establish relationships between transformer depth and automata composition.

From an architectural perspective, the work offers a detailed analysis of transformer components, including self-attention mechanisms, feed-forward networks, position encodings, and layer normalization effects. This analysis provides a precise characterization of computational capabilities and establishes clear mappings between architectural features and theoretical bounds.

The paper's impact extends beyond transformer analysis. By introducing a new complexity class for transformer computation, it bridges neural and classical computation models. The mathematical tools developed combine multiple theoretical frameworks effectively and create new connections between previously disparate domains. This work provides the first rigorous explanation of why CoT prompting is effective and establishes fundamental limits of transformer computation.

This work represents a significant advance in our theoretical understanding of transformer computation and CoT prompting. It provides a solid foundation for future research in neural architecture analysis and prompting strategies. The paper's thorough mathematical treatment and clear theoretical insights make it a landmark contribution to the field of machine learning theory.

Looking ahead, this work opens several promising research directions, particularly in understanding the optimal use of CoT prompting and the fundamental limitations of transformer architectures. The theoretical framework established here will likely serve as a foundation for analyzing future neural architecture innovations and prompting strategies.

https://arxiv.org/abs/2402.12875

Mike's Daily Paper - 11.01.25
Evaluating the Design Space of Diffusion-Based Generative Models

This paper provides a comprehensive analysis of diffusion-based generative models by offering a unified framework that bridges the training and sampling stages. It builds a robust mathematical foundation for understanding how design choices influence both model performance and computational efficiency.

It tackles the intricate interplay between training and sampling processes in diffusion models. Unlike previous work that often isolates these stages, this study provides a **unified error analysis** that integrates both. Here are the key contributions and insights:

Below, we delve into the main contributions:

1. Training Dynamics and Convergence Analysis

The paper rigorously examines the denoising score matching (DSM) objective within the framework of gradient descent. Using semi-smoothness techniques, it establishes exponential convergence for deep ReLU networks and provides insights into the optimal weighting functions for training. The focus on variance exploding (VE) settings is particularly notable, as it aligns with practical implementations like EDM (Karras et al., 2022).

Key insights into training dynamics include:

- The **bell-shaped weighting function** emerges naturally from the analysis. This weighting ensures that the optimization focuses more on intermediate noise levels, where the signal-to-noise ratio is balanced, making it easier for the neural network to learn accurate score functions.
- The derived gradient bounds rely on carefully designed assumptions about data scaling and input dimensionality, reflecting realistic training scenarios. These bounds not only guarantee convergence but also allow flexibility in network architectures and noise schedules.

By translating the theoretical findings into practical recommendations, the study emphasizes that the choice of weighting in the loss function is crucial for ensuring rapid convergence without compromising the generalization of the learned score.

2. Sampling Process and Error Bounds

The sampling process in diffusion models relies heavily on accurately simulating the reverse stochastic differential equation (SDE). The paper extends previous works by deriving sharper, non-asymptotic error bounds under general time schedules. This analysis covers initialization error, discretization error, and score approximation error.

Key contributions include:

- The complexity of the sampling process is shown to be **almost linear in data dimensionality**, provided optimal time schedules are used. This result has profound implications for the scalability of diffusion models, particularly in high-dimensional applications like image generation.
- The study evaluates the impact of time and variance schedules on sampling efficiency. It highlights how different schedules (polynomial vs. exponential) trade off between error minimization and computational cost, offering clear guidelines for different training scenarios.

The explicit breakdown of errors provides practitioners with actionable insights into tuning the generation process. The work also sheds light on the significance of noise initialization and its impact on the final sample quality, connecting theoretical error bounds with practical outcomes.

3. Full Error Analysis

By combining the training and sampling analyses, the authors develop a holistic framework for end-to-end error quantification in diffusion-based generative models. This integration reveals how various error sources interact and provides a unified view of the factors influencing sample quality.

Highlights of the full error analysis include:

- **Optimization Error Decomposition:** The study distinguishes between training-related errors (optimization and statistical errors) and sampling-related errors (discretization and initialization). This decomposition clarifies the interplay between model training and the generative process.
- **Impact of Model Overparameterization:** The results show how increased network width and depth can mitigate optimization errors, allowing gradient descent to achieve exponential convergence. This aligns with empirical observations in deep learning but provides a rigorous theoretical underpinning.
- **Error Bound Tightness:** The derived error bounds depend on key parameters such as data dimensionality, noise schedule, and weighting functions. For practical noise schedules (e.g., EDM), the bounds align closely with empirical performance metrics.

The analysis also underscores how errors propagate across training and sampling stages, offering insights into how to balance computational effort between these phases for optimal generative performance.

Appendix:
What is Semi-Smoothness?

Semi-smoothness is a property of the loss function and its gradient, ensuring that the gradient descent steps effectively decrease the loss even when the function is not perfectly smooth. For deep ReLU networks, the loss function involves piecewise linearities, making it non-smooth in general. The semi-smoothness property guarantees that:

- The gradient provides a meaningful direction for descent despite non-smoothness.
- There exist lower bounds on the gradient norms, ensuring consistent progress toward minimizing the loss.

By leveraging semi-smoothness, the authors establish a mathematical link between the loss value and the magnitude of its gradient, enabling them to prove exponential decay in the optimization error.

https://arxiv.org/abs/2406.12839

Mike's Daily Paper: 14.01.25
Improve Mathematical Reasoning in Language Models by Automated Process Supervision

I've long wanted to write a review about MCTS (Markov Chain Tree Search), and by complete coincidence, I came across this paper proposing to implement this cool method for training LLMs. This time, the goal is to train a language model to solve complex mathematical (logical) problems whose solutions contain many steps.

First, a brief explanation of what MCTS is. Monte Carlo Tree Search (MCTS) is an algorithm for policy optimization in finite-horizon and finite-size Markov Decision Processes, based on sampling random episodes organized through a decision tree.

It works in 4 stages:

- Selection: Choose a path from root to leaf according to an exploration/exploitation policy
- Expansion: Add a new state to the tree
- Simulation: Run a random simulation from the new state until the end of the game
- Backpropagation: Update the values in all nodes along the chosen path

MCTS is usually used to improve the policy by selecting better actions. The model provides state evaluations instead of random simulations, and MCTS uses these evaluations to build a more efficient search tree. For example, AlphaGo uses MCTS combined with deep networks to choose moves. The main advantage of MCTS is balancing between exploring new states (exploration) and utilizing existing knowledge (exploitation), improving decision-making over time.

The paper I'm reviewing today proposes using the MCTS approach to train a language model to build multi-step answers, and as you might guess, the nodes in this graph will be the solution steps. The paper notes that SOTA solutions for training language models to solve these problems fall into two types. The first simulates all solution steps so that the model is trained (with RLHF techniques of your choice) to maximize the reward the model receives at the end (usually binary, meaning whether the solution is correct/incorrect) with some regularization term (proximity to the original model).

The second method, called PRM, does something similar but for partial solution paths (=some solution steps at the beginning). We can see that the first approach will work less well for complicated problems with many steps because the reward is very sparse and difficult to optimize. The second case requires a lot of high-quality labeled data, which is very expensive.

The paper, as mentioned, proposes using MCTS together with PRM. As is common in MDP, we need to define what is the state, action, and reward. The state s is defined as the question q, all solution steps so far (doesn't have to include the whole solution), and action a is choosing the next node which in this case is the next step in solving question q. After action a is chosen, it's added to s, meaning the new state is (s_old, a). Action a is chosen by policy (p(a|s)) where for MCTS it consists of two terms: the first (exploitation) tends to choose nodes with high reward and the second term (exploration) prefers nodes we haven't visited much.

Now it's time to talk about the reward. For a given node v, its reward is the percentage of correct rollouts (denoted as c) that started from stage v (percentage of paths in the tree that reached the correct solution starting from v). By the way, there's a very intuitive method for identifying the first error in an incorrect solution (which several previous works found as effective information for training an LLM with PRM) that allows identifying nodes that are "definitely incorrect" (from which the correct solution cannot be reached) in the solution. The method is called "binary search."

The method each time divides the solution path in half and checks if c for the node found in the middle of the path is greater than or equal to 0. If it equals 0, then the error is probably in the first half, and if it's greater than 0, then the error is probably in the second half. Then again, divide in half the half where we suspect there's an error and continue to narrow the search until reaching the "misleading node."

To increase the number of examples, the authors propose storing solution rollouts and performing binary search of the node where (likely) an error occurred and starting a new search from there. This allows building examples with the same initial steps but different continuations. I'll remind you that with the PRM approach (which the paper builds its solution on), each example is a triplet of (question, partial solution, whether it's correct). All these quantities can be derived using techniques described above in this review

Finally, the paper uses MCTS with Q policy where the state of each node in the solution graph is described by a (different) triplet which consists of the number of times the solution visited this node, the percentage of correct solutions c from this node (i.e., its Monte Carlo estimator), and also a policy value Q that receives a high value for a c value close to 1 (node mostly leads to correct solution) and also has a (multiplicative) regularization term penalizing it for longer solutions. Selection of a rollout path is chosen by sampling built based on the tree statistics with an algorithm called PUCT (formula 3 in the paper). Of course, Q, c, and tree statistics are updated during MCTS.

That's it - a very long review, I hope I managed to explain it, not a trivial paper... https://arxiv.org/abs/2406.06592

Mike’s Daily Paper: 16.01.25
Diffusion Models for Non-autoregressive Text Generation: A Survey

Today we'll review a survey from a year and a half ago about a field (family of techniques) so naturally it (the review) will be quite brief. The survey is about non-autoregressive text generation methods, meaning not token after token but rather an entire sequence. The methods we'll discuss generate text in several iterations, but this isn't done in an autoregressive way - for example, these methods can generate token number 78 before token number 24.

Okay, many of you probably thought about generative diffusion models after I mentioned iterative methods, and you're not wrong here. In this brief review, I'll explain concisely how text can be generated using diffusion models. As you probably remember, diffusion models are trained to remove noise from noisy data through iterations. In other words, the model is trained to remove small amounts of noise from the data until reaching clean data, and thus after training, the model can generate data from pure noise in several iterations.

But how can we add noise to text that lives in a discrete space (i.e., tokens)? There are two broad approaches: the continuous approach and the discrete approach. In the continuous approach, which is similar to the standard generative diffusion models, we don't operate in discrete space but in the embedding space. In the continuous approach, we transform our text into a continuous embedding vector, but unlike a regular encoder, we transform each token into its vector representation separately from the others(non-contextualized embedding). Then we train a diffusion model to generate embeddings of texts similar to the way latent diffusion models are trained. It means that noise addition and denoising model training occur in the embedding space, with the ultimate goal being to predict the tokens from the embeddings generated by the latent diffusion model after noise cleaning.

The second family of methods is to perform noise addition in the discrete space. Obviously, the noise can't be continuous, so what can be done is to change token values (for example, to [mask] token) with a certain probability, with the goal being to turn all tokens into [mask] in the final iteration. A diffusion model at iteration i is trained to predict the tokens from the previous iteration, whereas during inference, generation starts with all tokens equal to [mask] and the model gradually turns them into text.

Of course, the “way tokens are noised” in each iteration is a hyperparameter equivalent to the noise schedule in regular diffusion models. It turns out that the noising method can be described by a matrix. Each token can be represented by a probability vector (over the token dictionary) so a token from iteration i can be represented as the inner product of its representation in iteration i-1 by a stochastic matrix Q_i (sum of rows and columns is 1). Q_i is the most important hyperparameter in discrete diffusion models.

It turns out this is quite an active research field although these models haven't yet reached the performance of autoregressive language models. But I'm not ruling out that this might still happen because these models can work at higher throughput than autoregressive models (for a modest number of iterations).

https://arxiv.org/abs/2303.06574

Mike's Daily Paper: 17.01.25
Towards a Unified View of Preference Learning for Large Language Models

Overview:

The paper tackles a critical challenge in the development of large language models (LLMs): aligning model outputs with human preferences. This alignment is essential for ethical, accurate, and user-friendly applications of LLMs. While reinforcement learning from human feedback (RLHF) and supervised fine-tuning (SFT) have been central to preference alignment, their relationships remain underexplored, leading to fragmented research.

The authors aim to unify these disparate efforts by introducing a framework that integrates RLHF and SFT approaches under a single gradient-based formulation. This unification not only bridges methodological gaps but also sets the stage for more cohesive advancements in preference learning. The paper emphasizes that preference alignment encompasses multiple components—model, data, feedback, and algorithm—each critical for robust performance.

Technical Contributions

Unified Gradient Framework

At the heart of the paper is the formulation of a unified gradient for preference optimization:

$$\nabla_\theta = \mathbb{E}_{(q,o)\sim D}\left[\frac{1}{|o|}\sum_{t=1}^{|o|}\delta A(r,q,o,t)\nabla_\theta \log \pi_\theta(o_t|q,o_{<t})\right],$$

Here:

- δ: A gradient coefficient that depends on the specific algorithm, feedback, and data.
- A The optimization algorithm applied.
- rr: Feedback signal influencing the gradient coefficient.
- πθ\pi_\theta: Policy model parameterized by θ\theta.

This equation generalizes the optimization processes used in both RL-based and SFT-based methods, showing that their core difference lies in how feedback is incorporated. RL-based methods typically use scalar rewards, while SFT employs preference labels or rankings.

Taxonomy of Preference Learning

The paper categorizes preference learning into four interconnected stages:

1. **Data**:

    - **On-Policy Data Collection**: Data is generated in real-time by the model being trained. Sampling techniques like Top-K, Nucleus Sampling, and Monte Carlo Tree Search (MCTS) are utilized for diverse and high-quality data generation.
    - **Off-Policy Data Collection**: Data is pre-collected, often from external sources, including human-annotated datasets (e.g., HH-RLHF, SHP) or synthetic datasets generated by LLMs (e.g., UltraChat, ULTRAFEEDBACK).

2. **Feedback**:
    - **Direct Feedback**: Includes human labels and hand-designed rules. Examples include correctness checks in mathematical reasoning or unit test results in code generation.
    - **Model-Based Feedback**:
        - **Reward Models**: Estimate human preference probabilities using methods like the Bradley-Terry model:

$$p^*(y_1 \succ y_2|x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}.$$

          Optimization is achieved through a negative log-likelihood loss:

$$L_r = -\log \sigma(r^*(y_c, x) - r^*(y_r, x)).$$

        - **Pairwise Scoring Models**: Focus on comparative ranking rather than absolute scoring. These are computationally lighter and more interpretable.
        - **LLM-as-a-Judge**: Utilizes LLMs themselves to evaluate outputs. Self-rewarding mechanisms, meta-rewarding, and generative verification extend this approach.
3. **Algorithms**: Algorithms are grouped based on the number of samples required for gradient computation:

    - **Point-Wise Methods**: Optimize using single data points. Examples include Proximal Policy Optimization (PPO) and ReMax.
    - **Pair-Wise Contrast Methods**: Leverage comparisons between pairs of outputs. Direct Preference Optimization (DPO) exemplifies this, with its loss function:

$$L_{\text{DPO}} = -\log \sigma \left(\beta \log \frac{\pi(y^+|x)}{\pi_{\text{ref}}(y^+|x)} - \beta \log \frac{\pi(y^-|x)}{\pi_{\text{ref}}(y^-|x)} \right).$$

    - **List-Wise Contrast Methods**: Evaluate and optimize over entire sets of outputs. This approach is particularly useful in tasks requiring holistic assessments, such as ranking or summarization.
    - **Training-Free Methods**: Include input/output optimization techniques, eliminating the need for gradient updates during alignment.
4. **Evaluation**: Evaluation strategies assess how well LLMs align with human preferences:

    - **Rule-Based Evaluation**: Uses predefined criteria such as factual correctness or task-specific benchmarks.
    - **LLM-Based Evaluation**: Involves advanced LLMs (e.g., GPT-4) acting as evaluators, using prompts to assess and rank responses.

Mathematical Details of Reward Models

The paper dives deep into reward modeling techniques, contrasting:

1. **Bradley-Terry Models**: Which rely on preference probabilities derived from pairwise comparisons.
2. **Binary Classification Models**: Simpler and applicable for tasks with binary outcomes (e.g., correctness checks in code generation or mathematical reasoning).

To address reward hacking and over-optimization, techniques like Kullback-Leibler divergence regularization, model ensembling, and fine-grained reward modeling are explored. For instance:

- Fine-grained RLHF assigns rewards to intermediate reasoning steps, improving performance in tasks requiring sequential decision-making.

Conclusion

This survey provides a mathematically rigorous and conceptually unified view of preference learning for LLMs. Its framework clarifies relationships between RL and SFT methods, enabling researchers to compare, combine, and innovate preference alignment strategies systematically. The emphasis on feedback, algorithm design, and evaluation ensures comprehensive coverage of the field, making this paper an invaluable resource for advancing LLM alignment research.

Mike's Daily Paper - 18.01.25
MAKING TEXT EMBEDDERS FEW-SHOT LEARNERS

Today, unlike recent reviews, we'll review a very light paper that doesn't involve heavy math. The paper proposes a method for building representations (embeddings) adapted for in-context learning, or ICL for short. Recall that ICL is a prompt-building method where we provide the model with several examples for a task we expect it to perform. For instance, in code generation tasks, we provide the model (within the prompt) with several examples, each being a pair of (question, code) to "clarify" to the model what we expect from it. By the way, why ICL sometimes

works on tasks the model wasn't trained on isn't 100% clear and is quite an active research topic.

Note that the model still needs to generate text, meaning we have a decoder model (with causal masking that quite interferes with building the embedding), and the question is how we build an embedding with it as we're used to doing with an encoder. By the way, several papers have proposed methods for building embeddings with decoder models like LLM2Vec and GritML, but they aren't adapted for the case discussed in this paper. That is, the question is how we build an embedding of an ICL-style prompt, meaning one that contains several solved examples for demonstration.

So the authors found a pretty simple solution for this. First, they added an EOS token at the end of the prompt, and the plan is that this token's representation will contain the prompt's embedding (as was done in BERT 7 years ago). Unsurprisingly, the authors chose to do this with contrastive learning (CL).

The goal of CL is to train a representation model so that representations of similar (positive) examples will be close while those of dissimilar (negative) examples will be far apart in the embedding space. For positive examples, the authors chose those with correct answers to the prompt's question, while for negative examples, the incorrect answer was chosen. Note that the demonstration examples in the prompt remain identical for both positives and negatives.

That's it - this is how they train an embedding model on a small number of examples (few-shot), and according to the paper, the results aren't bad…

https://arxiv.org/abs/2409.15700

Mike's Daily Paper - 18.01.25

A joint review by Orel Lavie and Mike of the legendary paper:

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

The main idea:

The Lottery Ticket Hypothesis states that within a dense neural network initialized randomly, there exists a subnetwork (or "winning ticket") that, when trained in isolation, can achieve performance comparable to the original network.

It was found that standard pruning techniques naturally reveal such subnetworks, for which reinitializing under the same hyperparameters preserves the results of the original network at a lower cost, such that the winning tickets are subnetworks that "won the initialization lottery," where the initial weights make training particularly effective.

This idea emphasizes the importance of the network's initial weights. The winning tickets are not random subnetworks, but rather ones that are especially suitable due to their initialization. The

process of finding these subnetworks is not simple, as it involves identifying the critical parts (significant neurons) in the network from the very beginning.

What is network pruning?

Pruning is a technique that removes unimportant weights from a neural network. According to the Lottery Ticket Hypothesis, pruning helps optimize the network by removing redundant neurons and connections, thus creating a lighter, faster, and more efficient network that maintains or sometimes improves upon the original network's performance. Pruning exposes the "winning tickets": initially, the network contains too many parameters (large and dense network), and then during training and pruning of insignificant weights, these efficient subnetworks are revealed.

Types of Pruning:

Unstructured Pruning: Here, any weight or group of weights can be removed without restrictions. This creates a "sparse" neural network where only some weights remain. This technique is also called Weight Pruning. In such pruning, there's no predefined choice of what will be pruned; it's all based on the minimal contribution of the chosen neuron to be pruned.

Structured Pruning: Here, entire groups of weights are removed, such as complete neurons in a feedforward network (FFN). The result is a "dense" but smaller neural network. The choice here is deliberate, where the network's structure is important to maintain. A neuron might not be chosen for pruning to avoid damaging the chosen structure, compared to other neurons.

One-shot versus Iterative Pruning:

One-shot Pruning: The network is trained once, a certain percentage of weights (p%) is pruned, and then the remaining weights are reinitialized. This assumes that in one iteration we've reached the final and desired solution, without need for a repeated and ongoing process.

Iterative Pruning: The network is trained, some weights are pruned, reinitialized, and the process is repeated several times. In each round, a small percentage of the weights that survived the previous round is pruned. Results show that iterative pruning successfully finds winning tickets that achieve the same performance as the original network, while using a smaller network compared to one-shot pruning.

https://arxiv.org/pdf/1803.03635

Mike's Daily Paper - 21.01.25
Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts

The paper caught my attention despite my shallow knowledge of time-series(TS) analysis. Mainly because its name includes the phrase "Foundation Models," which is quite a rare beast

in the time-series field, unlike in large language models. The reason for this (probably) is the much richer variety of time series that are relatively different compared to natural language.

To be honest, I didn't find very interesting architectural innovations in Time-MoE, which is of course based on transformers, however, it has some elements that are different from what we're used to seeing in LLMs. For instance, instead of tokenization and embedding layers based on a token dictionary that we have in LLMs, in the proposed model each token (which is a point in the series) undergoes a non-linear transformation with SwiGLU activation and several linear transformations.

Regarding the transformer layer, the authors use a fairly standard MoE architecture. The only difference that caught my eye was the use of the RMSNorm normalization method that I wasn't familiar with. Besides that, it has all the regular transformer layers including, of course, residual layers.

The final layer of Time-MoE is a bit different from what we're used to seeing in transformers. Since unlike language models, in the TS world we need to predict different time points (say a second, minute, or day ahead), the authors use multiple heads in the last layer. Each head is responsible for predicting at a specific horizon (number of samples ahead). During training, they combine the losses from all heads.

The loss functions in the paper are quite standard too: Huber function, which is the robust version of L2 (doesn't allow reaching very high values). Additionally, there's a regularization term that tries to activate all experts in MoE uniformly. And of course, they trained the model on huge and diverse datasets.

That's it - a short review, and hopefully a clear one too…

https://arxiv.org/pdf/2409.16040

Mike's Daily Paper - 22.01.25
MONOFORMER: ONE TRANSFORMER FOR BOTH DIFFUSION AND AUTOREGRESSION

Today I'll do a brief review of a quite interesting paper that combined two types of models - a language model and a vision model into a single transformer. Most multimodal models consist of

several models, each responsible for generating one type of data. For example, visual language models typically consist of two models: a language model and an image generation model. The authors propose to "connect" these two models into a single transformer model, and this is done quite intuitively.

First, let's note that both these models work in the token space, where for language models each token is part of a word or a complete word, while for visual models each token is an image patch. So the attempt to combine them into one model seems quite natural, but it's unclear whether the same transformer can be trained to generate both language and images.

The proposed model generates language exactly like a standard LLM, in an autoregressive manner, meaning token after token. But how can it be combined with an image generation model that is obviously based on diffusion models (in 2025 this is the default option after all). First, we need to remember that an autoregressive model (for language generation) works causally, meaning that during the generation of token n, all tokens behind it are masked and don't participate in generation (using a causal mask). For diffusion models, we need a bidirectional model because when generating an image patch, it's very beneficial to use all other patches.

This is exactly how the proposed model is built - language is generated with a causal mask and the image is generated with all tokens (including text tokens). By the way, this approach will also work in the opposite direction: meaning for generating text from an image (for example, for the image captioning task). But how do we know to switch from "causal" mode to "bidirectional" mode? The authors suggest using a specific token that marks where image generation begins - this token should be generated for tasks like text-to-image generation.

A few words about the transformer for image generation. The paper uses a latent diffusion model where the model is trained to build a latent representation of an image from noise by gradually removing a noisy component from it (for each patch). Then all representations (of the patches) are passed through a decoder (based on VAE) that builds an image from them.

The model is trained with a loss that is a weighted sum of the standard losses for the above-mentioned models: language model and diffusion model. The paper manages to generate quite beautiful images....

https://arxiv.org/abs/2409.16280

Mike's Daily Paper - 24.01.25
Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs

Short Paper Summary:
The paper revisits the use of Reinforcement Learning from Human Feedback (RLHF) in the optimization of Large Language Models (LLMs). It challenges the dominance of Proximal Policy Optimization (PPO) as the de facto reinforcement learning(RL) method in this context, highlighting its computational inefficiency and unnecessary complexity. Instead, the authors propose returning to simpler REINFORCE-style methods, specifically Vanilla Policy Gradient (REINFORCE) and its multi-sample extension, REINFORCE Leave-One-Out (RLOO). These methods are shown to outperform PPO in terms of computational cost, sample efficiency, and reward optimization across multiple datasets and LLM architectures. The findings emphasize that aligning LLMs with human preferences can be achieved with more straightforward optimization strategies tailored to the specifics of RLHF.

Important points:

Theoretical Simplification:
The authors demonstrate that many components of PPO (e.g., clipping, value functions, and token-level modeling) are unnecessary for RLHF, given the stable initialization(aka warm start) of pre-trained LLMs. By modeling entire sequences as single actions, REINFORCE avoids the complexity of token-level state-value functions, making the problem akin to a contextual bandit.

Practical Efficiency:
RLOO uses all generated samples for baseline construction, achieving higher sample efficiency than RAFT, which discards all but the top-ranked sample. This leads to significant computational savings and better utilization of available data. The approach simplifies RLHF pipelines by reducing reliance on sensitive hyperparameters like clipping ratios and advantage estimation.

Robustness:
RLOO demonstrates robustness to noisy reward signals and higher KL penalties, outperforming methods like RAFT that are more sensitive to them.

Theoretical Insights

Variance-Bias Tradeoff and Unbiased Gradient Estimation:
PPO relies on state-value functions and Generalized Advantage Estimation (GAE) to reduce variance in gradient estimation at the cost of introducing bias. The paper argues that in RLHF, the strong initialization(=warm start) of LLMs makes variance reduction less critical. This enables unbiased methods like REINFORCE to perform well without introducing bias. Empirically, the paper demonstrates that REINFORCE achieves better reward optimization than PPO, even when subjected to theoretically high-variance conditions.

Full-Trajectory Modeling vs. Token-Level Modeling:
PPO models each token as an action, creating a Markov Decision Process (MDP) where partial sequences are states. However, RLHF attributes rewards only to entire sequences, rendering intermediate states irrelevant. By modeling the entire sequence as a single action, REINFORCE simplifies the problem into a bandit-like setup, aligning directly with the reward structure. Empirical results confirm that this approach outperforms token-level modeling in both efficiency and performance.

Clipping and Stability of Policy Updates:
PPO uses a clipping mechanism to prevent large policy updates that could destabilize learning. The authors show that this is unnecessary for RLHF, as the optimization landscape is stable due to the strong initialization of pre-trained LLMs. Removing clipping in PPO or avoiding it altogether with REINFORCE results in better performance, indicating that RLHF does not require this level of stabilization.

Tradeoff Between Variance Reduction and Bias Introduction:
PPO's advantage estimator trades off variance and bias, controlled by the hyperparameter λ. Higher λ\lambda values (closer to 1) reduce bias but increase variance. The authors demonstrate that in RLHF, higher λ\lambda values consistently lead to better rewards, supporting the use of unbiased estimators like REINFORCE.

Robustness of Multi-Sample Baseline Estimators:
RLOO uses multiple generated samples to construct baselines for variance reduction. This approach maintains the unbiased nature of REINFORCE while significantly improving sample efficiency and robustness to noise. Unlike RAFT, which discards lower-ranked samples, RLOO leverages all generated samples, leading to consistent improvements across datasets and conditions.

KL Regularization and Alignment:
KL divergence penalizes deviations from the reference policy, helping ensure alignment. The authors show that RLOO handles high KL penalties better than RAFT, which struggles due to its reliance on top-ranked samples. RLOO achieves a balance between alignment (reward optimization) and diversity, avoiding excessive overfitting to the reward model.

Simplification of RLHF Objectives:
By eliminating components like clipping, token-level modeling, and GAE, the REINFORCE framework reduces the number of hyperparameters, simplifying the RLHF pipeline. This makes it more accessible to non-RL specialists while maintaining or improving performance.

Limitations and Future Directions:

Reward Over-Optimization:
The study does not address reward model over-optimization, where the policy exploits biases in the reward function at the expense of generalization. This remains an open challenge for RLHF.

Human Evaluation:
While simulated win-rates using GPT-4 serve as a proxy for human preferences, direct human evaluations would provide stronger evidence for the alignment quality.

Scalability:
The scalability of REINFORCE and RLOO to larger models and diverse datasets, particularly under resource constraints, warrants further investigation.

Conclusion
The paper presents a compelling argument for revisiting REINFORCE-style methods in RLHF, challenging the dominance of PPO. By leveraging the specific characteristics of RLHF—such as stable pre-trained initialization and sequence-level rewards—the authors demonstrate that simpler methods like REINFORCE and RLOO can outperform more complex alternatives like PPO and RAFT in terms of reward optimization, sample efficiency, and robustness.

https://arxiv.org/abs/2402.14740

-

Mike's Daily Paper Review - 27.01.25
FineZip: Pushing the Limits of Large Language Models for Practical Lossless Text Compression

I chose this Paper for review because I have a great fondness for everything related to compression - whether it's model compression, data compression, or any kind of compression :). The Paper proposes a nice method for data compression. You probably know that our models can compress data quite well with the latent representation (embedding) they produce from the data. If I'm not mistaken, we can compress a high-resolution image by a factor of 100 with its embedding.

But this compression isn't lossless. While we can reconstruct the image from its embedding so that the human eye won't notice any difference between the reconstructed and original images, they aren't necessarily identical. In the case of text, this can be somewhat problematic because we want to recover it exactly as it is.

The reviewed paper, however, proposes a method for text compression that allows for exact reconstruction. The proposed method is quite simple and intuitive. After all, how does a language model generate text - its final layer outputs a distribution over the token space and the token is sampled from this distribution (there are several methods). We can use this distribution to compress our text.

For example, as proposed in the LLMZip paper, we can encode each token (given its preceding context) by the rank of its probability in the distribution for that token. This rank is essentially the position of the token in the sorted list of tokens according to their probability in that token's distribution(in descending order).

If the language model we're using is very powerful, this rank will be close to 1 (or 2, 3 but not 1000). And it's known that such sequences can be compressed very efficiently (high compression rate). So LLMZip proposed fine-tuning a language model for the text being compressed, which isn't practical because each text needs its own model and the training is computationally intensive. The reviewed paper suggests using LoRA (or other PEFT method) for compression so that we'll need to store the additional matrices (or adapters) for each text with a single language model for all.

While it's still not very practical, it's quite interesting from a conceptual perspective..

https://arxiv.org/abs/2409.17141

Mike's Daily Paper Review - 29.01.25
A Survey on Diffusion Models for Inverse Problems

Diffusion models have rapidly emerged as a powerful tool for generative modeling, capable of producing high-quality samples across diverse domains. Their success has paved the way for groundbreaking advancements in solving inverse problems, particularly in image restoration and reconstruction, where diffusion models serve as unsupervised priors.

The survey, I am going to review today, offers a comprehensive exploration of methods leveraging pre-trained diffusion models to address inverse problems without the need for additional training. The authors present a structured taxonomy that categorizes these approaches based on the specific problems they tackle and the techniques they employ.

Mathematical Framework for Inverse Problems

The paper formalizes inverse problems under the general formulation:

$$Y = A(X) + \sigma_y Z, \quad Z \sim \mathcal{N}(0, I_m)$$

where A is a (possibly nonlinear) corruption operator, and controls the noise level, X is an original clean data and Y is corrupted data. Various well-known problem settings, such as denoising, inpainting, and compressed sensing, are framed within this formulation by specifying different forms of A.

2. Diffusion Processes

The authors leverage **Denoising Diffusion Probabilistic Models (DDPMs)** and their extensions based on stochastic differential equations (SDEs) to approach inverse problems. The forward process is described by:

$$\mathrm{d}X_t = f(X_t, t) \mathrm{d}t + g(t) \mathrm{d}W_t,$$

where W_t is a Wiener process, X_t is data distribution at iteration(time) t. f and g are hyperparameters of the diffusion process(noise schedule). Anderson's reverse Stochastic Differential Equations(SDE) framework is used to sample from the unknown data distribution:

\mathrm{d}X_t = \left(f(X_t, t) - g^2(t) \nabla_{X_t} \log p_t(X_t) \right) \mathrm{d}t + g(t) \mathrm{d}W_t.

This formulation enables modeling corrupted data by progressively adding noise and subsequently reversing the diffusion process for reconstruction. The core mathematical challenge is estimating the **score function** which is the gradient of the probability distribution p_t(x_t). The survey highlights the pivotal role of **Tweedie's formula**:

\nabla_{x_t} \log p_t(x_t) = \frac{\mathbb{E}[X_0 | X_t = x_t] - x_t}{\sigma_t^2}.

Learning the conditional expectation using neural networks provides an effective way to approximate the score.

4. Taxonomy of Methods in Diffusion-Based Inverse Problem Solving

The authors of the paper provide a rich taxonomy that categorizes methods based on their mathematical approach, target problem types, and optimization techniques.

4.1 Explicit Approximations for Measurement Matching

This family focuses on approximating the measurement score term in a diffusion framework. These approximations often leverage closed-form solutions for linear inverse problems. The general form is given by:

\nabla_{x_t} \log p(y | x_t) \approx -L_t M_t G_t^{-1},

where:

- represents the measurement error, typically defined as .
- projects the error back into the solution space; for example, in linear cases.
- is a re-scaling factor controlling guidance strength.

Representative Methods

- **Score-ALD (**ALD stands for Annealed Langevin Dynamics)**:** Uses a simple approximation:
  \nabla_{x_t} \log p(y | x_t) \approx -\frac{A^T (y - A x_t)}{\sigma_y^2 + \gamma_t^2}.
- **DPS (Diffusion Posterior Sampling):** Approximates the posterior using a mapping:
  p(y | X_0 = \mathbb{E}[X_0 | X_t]) \sim \mathcal{N}(y; A \mathbb{E}[X_0 | X_t], \sigma_y^2 I),
  leading to:
  \nabla_{x_t} \log p(y | x_t) \propto A^T (y - A \mathbb{E}[X_0 | X_t]).

- **Moment Matching:** Extends DPS by incorporating an anisotropic Gaussian approximation:
  p(x_0 | x_t) \approx \mathcal{N}(\mathbb{E}[X_0 | X_t], \sigma_t^2 \nabla_{x_t} \mathbb{E}[X_0 | X_t]).

4.2 Variational Inference Methods

These methods approximate the true posterior distribution by introducing a tractable surrogate distribution and optimizing its parameters via variational techniques. The goal is to minimize the KL divergence between the surrogate and true posterior:

\min_{q} D_{KL}(q(x) \| p(x | y)).

Key Techniques:

- **RED-Diff:** Proposes a novel loss combining reconstruction and score-matching objectives:
  \mathcal{L}_{\text{RED}}(\mu) = \frac{1}{2 \sigma_y^2} \| y - A \mu \|^2 + \sum_t \lambda_t \| \epsilon_\theta(x_t) - \epsilon \|^2,
  where is the variational mean, and is the denoising function learned by the diffusion model.
- **Blind RED-Diff:** Extends RED-Diff by jointly optimizing for both the latent image and forward model parameters . This leads to a coupled variational problem:
  \min_{q} D_{KL}(q(x, \phi) \| p(x, \phi | y)).

4.3 CSGM-Type Methods (Conditional Score Generative Models)

These approaches optimize directly over a latent space using backpropagation. The fundamental idea is to iteratively adjust initial noise vectors to satisfy measurement constraints.

Key Techniques

- Backpropagation through a deterministic diffusion sampler.
- Latent space optimization to enforce fidelity to the observed measurements.

4.4 Asymptotically Exact Methods

These methods rely on sampling from the true posterior distribution using advanced Markov Chain Monte Carlo (MCMC) techniques.

Key Techniques

- **Particle Propagation:** Sequential Monte Carlo (SMC) methods propagate multiple particles through distributions to approximate the posterior.
- **Twisted Sampling:** Methods like the Twisted Diffusion Sampler (TDS) use geometry-aware updates to enhance convergence rates.

4.5 Optimization Techniques

The methods further vary by the optimization strategies employed:

- **Gradient-based techniques:** Utilize derivatives to enforce measurement consistency.
- **Projection-based techniques:** Project samples onto feasible subspaces.
- **Sampling techniques:** Use probabilistic approaches like Langevin Dynamics for particle updates.

This survey elegantly brings together advanced mathematical tools, providing a solid foundation for researchers aiming to solve inverse problems using diffusion processes. The integration of stochastic calculus, Bayesian inference, and optimization techniques makes it a crucial reference point for pushing the boundaries of inverse problem-solving.

https://arxiv.org/pdf/2410.00083

Mike's Daily Paper - 31.01.25
Law of the Weakest Link: Cross Capabilities of Large Language Models

Introduction and Problem Definition

The authors highlight a critical gap in existing LLM research - the tendency to focus on evaluating isolated capabilities while overlooking real-world tasks that require multiple skills, termed *cross capabilities*. The paper frames this problem through a comprehensive taxonomy of seven individual and seven cross capabilities, such as *Coding & Reasoning* and *Tool Use & Coding*. To address the complexity inherent in evaluating these intersections, the authors propose CrossEval, a benchmark composed of 1,400 human-annotated prompts designed to test LLM performance on multi-dimensional tasks.

Contributions and Methodology

The authors make several significant contributions:

Comprehensive Capability Definitions: They construct a detailed taxonomy of both individual and cross capabilities, categorizing tasks into broad categories and fine-grained subcategories.

CrossEval Benchmark: This novel evaluation framework consists of 1,400 prompts, 4,200 model responses, and 8,400 human ratings. The prompt set encompasses tasks of varying difficulty, ranging from simple factual questions to complex, cross-capability tasks.

Cross Capability Examples:

1. **Coding & Reasoning:** One prompt in this category may ask the model to analyze a code snippet and determine whether it correctly implements a complex mathematical function. This task requires not only coding knowledge but also logical reasoning to validate the function's correctness.

2. **Tool Use & Reasoning:** In another example, a prompt might require the model to use web-based data retrieval tools to answer a question about historical weather trends, followed by providing an analytical step-by-step explanation of observed patterns. This task demands both reasoning and external tool usage capabilities.

LLM-Based Evaluation: The study introduces a multi-reference evaluation framework where expert annotators assess the quality of multiple model responses on a Likert scale. The authors also develop a point deduction-based evaluation strategy for enhanced accuracy.

Analysis of Cross Capability Dynamics: The authors find that cross-capability performance often follows the "Law of the Weakest Link" — where performance is constrained by the weakest individual capability.

Experimental Findings

Law of the Weakest Link:
The most striking observation is that cross-capability performance is constrained by the weakest individual capability, consistent with the "Law of the Weakest Link." Out of the 58 cross-capability scenarios tested across 17 LLMs, 38 showed performance lower than either of the involved individual capabilities, while 20 scores lay between the strong and weak abilities but were much closer to the weaker one. For instance, in tasks combining Tool Use & Reasoning, if the model exhibited poor reasoning skills, it significantly degraded performance even when the model's ability to use tools was proficient. This effect was observed regardless of the complexity or nature of the task.

Tool Use Deficiencies:
Tool Use emerged as the weakest capability across all LLMs tested. Tasks requiring web browsing, dynamic data retrieval, or external code execution proved especially challenging. The highest scores for tasks involving tool use never exceeded 50 on a 1–100 scale across the benchmark. Notably, even models with code interpreter functionalities, such as Gemini Pro Exp, struggled to maintain performance parity with simpler reasoning tasks. This weakness is critical because tool use is fundamental to many real-world applications, such as research assistance, data analysis, and AI agents. The authors highlight that models relying solely on static data sources performed poorly compared to tasks where more explicit information was available directly within the prompt.

Cross-Capability Performance Gap:
On average, models scored 65.72 for individual capability tasks but only 58.67 for cross-capability tasks, a gap of 7.05 points. This highlights the difficulty models face when integrating multiple skills. Tasks such as Spanish & Reasoning and Long Context & Coding demonstrated particularly large gaps, suggesting that further optimization is needed in multilingual and long-context processing scenarios.

CrossEval Effectiveness in Differentiation:
CrossEval proved effective at distinguishing between even subtle differences among state-of-the-art LLMs. For example, the Claude 3.5 Sonnet model consistently outperformed its predecessors in tasks involving Image Recognition & Reasoning and Spanish & Image Recognition. This progression mirrors the development of increasingly sophisticated Claude models and emphasizes the value of CrossEval in benchmarking fine-grained advancements in LLM capabilities.

Correlation Metrics Improvement:
The benchmark demonstrated an improvement in correlation metrics for LLM-based evaluations when using multiple reference examples, given to the LLM, performing evaluation. Pearson correlation improved from 0.578 with no reference examples to 0.697 with two references, indicating that the inclusion of well-annotated references significantly enhanced evaluation reliability.

Summary:
The experiments reveal that while LLMs are advancing rapidly, they remain highly constrained by their weakest components. Addressing these limitations is essential for achieving more robust, cross-functional AI systems capable of solving complex, real-world problems.

https://arxiv.org/abs/2409.19951

Mike's Daily Paper - 31.01.25
Classical Statistical (In-Sample) Intuitions Don't GeneralizeWell: A Note on Bias-Variance Tradeoffs, Overfitting and Moving from Fixed to Random Designs

Introduction

Modern machine learning methods exhibit behaviors that starkly contradict traditional statistical intuitions, particularly concerning overfitting, the bias-variance tradeoff, and generalization. Classical statistics often posits that as model complexity increases, bias decreases, but variance increases—a well-known bias-variance tradeoff. However, phenomena such as *double descent* and *benign overfitting* challenge this view. The paper argues that these phenomena are not exclusively due to complex models, overparameterization, or high-dimensional data, but rather to a fundamental shift from *fixed design* to *random design* setups. The paper provides a fascinating mathematical exploration of how this shift significantly alters statistical principles.

Problem Setup: Fixed vs. Random Design

The distinction between fixed and random designs is fundamental to the paper:

- **Fixed Design:** The inputs during testing remain the same as the training inputs, *with only the labels re-sampled*. Classical statistical analysis often assumes this setup, which emphasizes minimizing *in-sample prediction error.*

- **Random Design:** *Both inputs and labels are independently sampled* from the underlying data distribution during testing. This setup aligns with how ML models are evaluated today, focusing on *generalization error or out-of-sample prediction error.*

The move from fixed to random design leads to profound changes in the behavior of bias, variance, and overall prediction error. This subtle yet impactful shift is the core reason why modern ML phenomena seem to violate classical statistical wisdom.

Mathematically, the errors in the two settings are defined as:

- **Fixed Design Error (In-Sample)**

  \text{ERR}_{\text{fixed}} = \mathbb{E}_{\tilde{y}} \left[\frac{1}{n} \sum_{i=1}^n (\tilde{y}_i - \hat{f}(x_i))^2 \right]

  where y_i^{~} are re-sampled outcomes at fixed inputs .

- **Random Design Error (Out-of-Sample)**

  \text{ERR}_{\text{random}} = \mathbb{E}_{x_0, y_0} \left[(y_0 - \hat{f}(x_0))^2 \right]

  where both x_0 and y_0 are new samples from the data distribution.

This simple change leads to far-reaching consequences for the bias-variance tradeoff and generalization properties of models.

Bias-Variance Tradeoff in Fixed vs. Random Design

Classical View (Fixed Design)

In classical fixed design setups, the bias-variance decomposition is straightforward:

\text{Bias}(x) = f^*(x) - \mathbb{E}[\hat{f}(x)]

\text{Var}(x) = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]

where:

- σ^2 is the irreducible noise in the data.

For simple estimators like -Nearest Neighbors (k-NN):

- **Variance** monotonically decreases with increasing as more neighbors are averaged.
- **Bias** monotonically increases since averaging incorporates less similar neighbors.

This tradeoff forms the textbook U-shaped curve for prediction error as a function of model complexity.

New Behavior in Random Design

Curth demonstrates that in random design, the bias-variance intuition breaks down. Specifically:

1. **Bias does not monotonically decrease with complexity:** The nearest neighbor may not perfectly match the test point, leading to non-zero *neighbor-matching bias*.
2. The bias can exhibit a *U-shaped pattern*, where intermediate complexity models minimize bias.

This behavior is formalized by decomposing the bias as:

\text{Bias}_k(x_0) = \left(f^*(x_0) - f^*\left(\sum_{i=1}^n w_{k,i}(x_0) x_i \right) \right) + \left(f^*\left(\sum_{i=1}^n w_{k,i}(x_0

The two components are:

- **Neighbor Matching Bias:** Arises when the weighted average of training points does not perfectly reconstruct the test point.
- **Averaging Bias:** Results from the nonlinearity of the true function f*(x).

This decomposition reveals that even in low-dimensional, simple settings, random design introduces complexities that disrupt classical intuitions.

Double Descent Phenomenon

Double descent refers to the non-monotonic behavior of prediction error as a function of model complexity. It consists of:

1. A U-shaped curve in the under-parameterized regime ().
2. A second descent in the overparameterized regime ().

Curth emphasizes that double descent cannot occur in fixed design settings because interpolation always results in constant in-sample error:

\text{ERR}_{\text{fixed}} = 2 \sigma^2 \quad \text{for any interpolating model}

This is because interpolating models predict perfectly at training points, leading to zero bias and variance in fixed design.

However, in random design, double descent emerges naturally due to changes in effective model complexity and generalization properties when moving beyond interpolation.

Benign Overfitting vs. Benign Interpolation

The author critiques the term *benign overfitting*, proposing *benign interpolation* instead. Classical definitions of overfitting imply degraded generalization performance, which contradicts the idea that fitting the training data perfectly can sometimes yield good test performance.

In fixed design, interpolation cannot be benign due to the dominance of noise variance:

\text{ERR}_{\text{fixed}} = \sigma^2.

In random design, however, models like neural networks and random forests can exhibit *spiked-smooth behavior*, where they interpolate sharply at training points but generalize smoothly to unseen inputs.

This behavior can be quantified using **effective complexity** measures: models that reduce effective complexity at test time tend to exhibit benign interpolation.

Experimental Validation

Curth provides empirical evidence using nonlinear data-generating processes (DGPs) to illustrate the theoretical findings:

1. **Bias Behavior:** In random design, bias exhibits a U-shape, contradicting the classical monotonic decrease.
2. **Double Descent:** Linear regression experiments confirm that double descent is exclusive to random design.
3. **Benign Interpolation:** Random forests behave like 1-NN at training points but generalize like -NN with test points.

Implications

The findings have significant implications for both theory and practice:

1. **Rethinking Statistical Education:** Introductory courses should clarify the distinction between fixed and random design settings.
2. **Causal Inference and ML:** In domains where training inputs may reoccur (e.g., causal inference), fixed design assumptions may still be relevant.
3. **ML Model Selection:** Understanding when interpolation is benign requires measuring test-time complexity, not just training performance.

Conclusion

Curth's work offers a groundbreaking perspective on why classical statistical intuitions break down in modern ML. By highlighting the shift from fixed to random design, the paper provides a unifying framework for understanding double descent, benign interpolation, and the evolving role of the bias-variance tradeoff.

This note invites statisticians and ML practitioners to reconsider long-held assumptions and adapt their methodologies to align with the realities of random design settings.

Mike's Daily Paper - 03.02.25
The Perfect Blend: Redefining RLHF with Mixture of Judges

Following the release of DeepSeek's latest model, there's increased interest in RLHF (Reinforcement Learning with Human Feedback) as a technique for fine-tuning language models. DeepSeek researchers demonstrated that it's possible to train a powerful language model to perform reasoning primarily using RLHF (there's some SFT but still mostly RLHF). The paper we'll review today was published almost 4 months before DeepSeek's R1 and proposes a method that improves RLHF performance.

One of the major problems with RLHF training is reward hacking (or RH), where the model learns to maximize the reward function but converges to a weak or unsafe model (despite the regularization term that tries to keep the final model close to the initial model used for RLHF). The authors propose to address this problem in three ways. The first is a set of constraints on the prompt response (such as whether it's harmful) checked by a judge (which is another language model). The second improvement is a modification of the reward function by subtracting a certain baseline reward, which I'll explain shortly. The third change is in building the dataset for RLHF.

This subtraction reminds me of two things. First, the new reward (after subtraction) looks similar to the advantage function (just similar but it's not) known to us from the PPO loss function, except this time it's not calculated through a Value function using GAE methods but in a different way. This new reward reminds us of what we saw in DeepSeek's paper's objective function. There, the baseline was calculated using the average reward (over batch) of the fine-tuned model (normalized with variance). In the paper, this term served as an estimate of that advantage function.

As mentioned, the paper's second innovation (the first being the constraints we impose on model outputs) is the baseline subtracted from the reward. The authors suggest taking the baseline as the reward for gold examples (answers) from the SFT dataset (questions and answers) or from preferred answers in the RLHF dataset. Thus, our reward measures how the trained model's answers compare to preferred answers in terms of their reward.

The third change is in the objective function. In addition (the authors propose 2 variants) to maximizing the likelihood of preferred answers and minimizing the likelihood of less preferred

answers (in terms of reward), the paper suggests only maximizing the likelihood of preferred answers (surprisingly, this works). The authors also "wrap" these ideas with classical RLHF methods like DPO and RAFT.

https://arxiv.org/abs/2409.20370

Mike's Daily Paper - 05.02.25
Technical Review: Deep Generative Models through the Lens of the Manifold Hypothesis

Overview:

Curious about why diffusion models outperform GANs, VAEs, and other generative approaches from a mathematical perspective? Want to understand the brilliance behind latent diffusion models and why they excel? Dive into this in-depth, technical, and mathematically rich paper review — a fascinating journey into the theory behind modern generative models!

This paper offers a comprehensive exploration of deep generative models (DGMs) under the framework of the manifold hypothesis, which asserts that high-dimensional data of interest often lies on a lower-dimensional submanifold embedded within the ambient space. The authors aim to clarify why models such as diffusion models and certain generative adversarial networks (GANs) empirically outperform others, including likelihood-based methods like variational autoencoders (VAEs) and normalizing flows (NFs). By adopting a manifold-based viewpoint, the authors provide insights into the intrinsic limitations of existing approaches while introducing new theoretical connections between DGMs and optimal transport.

The study stands out by formally proving the inherent numerical instability faced by high-dimensional likelihood-based models when attempting to represent manifold-supported data and establishing a novel interpretation of two-step DGMs as approximators of the Wasserstein distance.

Key Contributions

1. Survey of Manifold-Aware and Manifold-Unaware DGMs

The authors categorize DGMs into two primary groups:

- **Manifold-Unaware Models:** These models do not explicitly account for the manifold structure of data. Examples include VAEs, NFs, and energy-based models. Such models are prone to manifold overfitting, where densities diverge to infinity along the manifold while failing to capture the distribution within it.
- **Manifold-Aware Models:** These models either incorporate noise to spread probability mass beyond the manifold or optimize support-agnostic objectives that implicitly capture

manifold structure. Examples include diffusion models, conditional flow matching, and Wasserstein GANs.

This categorization helps elucidate why manifold-aware models often outperform their unaware counterparts in generating high-quality samples.

2. Numerical Instability of Likelihood-Based Methods

One of the central theoretical contributions is the proof that likelihood-based models suffer from unavoidable numerical instability when modeling manifold-supported data. The authors demonstrate that as model densities attempt to concentrate on the manifold, the likelihood function becomes unbounded, leading to degenerate solutions.

Mathematically, let pXp_X be the data distribution supported on a manifold M⊂RDM \subset \mathbb{R}^D with intrinsic dimension d∗d^*. For any sequence of likelihood-based models {pX,θt}\{ p_{X, \theta_t} \} approximating the data distribution, it holds that:

\lim_{t \to \infty} p_{X,\theta_t}(x) = \infty \text{ for all } x \in M.

This result implies that full-dimensional densities inherently diverge when tasked with modeling manifold-supported distributions, making likelihood-based objectives ill-posed for such data.

3. Limitations of KL Divergence

The authors emphasize that KL divergence, a commonly used objective for training DGMs, becomes ineffective in the manifold setting. The primary issue arises because KL divergence assumes that both distributions share the same support. However, when comparing a full-dimensional model density p_{X,θ} with a manifold-supported data distribution p_X, the KL divergence becomes infinite:

KL(p_X|||p_{X,\theta}) = \infty.

This divergence occurs because pX assigns non-zero probability only to points on the manifold, whereas p_{X,θ} spreads probability mass across the entire ambient space. Consequently, likelihood maximization, which is equivalent to minimizing the KL divergence, fails to provide a meaningful learning signal.

4. Wasserstein Distance as an Alternative Objective

To address the limitations of KL divergence, the authors advocate for the use of Wasserstein distances, which remain well-defined even when distributions have mismatched supports. The Wasserstein-1 distance between two distributions p and q is defined as:

W1(p,q)=infγ∈Π(p,q)E(X,Y)~γ[‖X−Y‖],W_1(p, q) = \inf_{\gamma \in \Pi(p, q)} \mathbb{E}_{(X, Y) \sim \gamma} [\| X - Y \|],

where Π(p,q)\Pi(p, q) denotes the set of joint distributions with marginals p and q. Unlike KL divergence, the Wasserstein distance measures weak convergence, making it a robust objective for training DGMs in the manifold setting.

5. Interpretation of Two-Step Models

The authors provide a novel interpretation of two-step DGMs, such as latent diffusion models, which first learn a low-dimensional representation of the data manifold and then model the distribution within this representation. They show that these models effectively minimize an upper bound of the Wasserstein distance between the model distribution and the true data distribution:

W(pX,pX,θ)≤Reconstruction Error+Distributional Divergence,W(p_X, p_{X, \theta}) \leq \text{Reconstruction Error} + \text{Distributional Divergence},

where the reconstruction error measures how well the learned manifold approximates the true data manifold, and the distributional divergence quantifies the difference between distributions within the learned manifold.

This result provides a theoretical justification for the empirical success of latent diffusion models and other two-step approaches.

Mathematical Insights

Numerical Instability Theorem

The authors formally prove that for any manifold-supported data distribution p_X and any sequence of full-dimensional model densities {p_{X,θ_t}} the likelihood objective does not admit a maximum. This result is derived by analyzing the behavior of densities on low-dimensional manifolds and leveraging properties of differential geometry and measure theory.

Wasserstein Distance Minimization

By framing two-step models as approximators of the Wasserstein distance, the authors establish a connection between manifold learning and optimal transport. This insight not only explains the superior performance of latent diffusion models but also provides a principled framework for designing new DGMs.

Failure of KL Divergence

The paper highlights the fundamental mismatch between manifold-supported and full-dimensional distributions, which leads to the divergence of KL-based objectives. This problem is formalized using the Radon-Nikodym derivative, which fails to exist when the supports of the distributions do not overlap.

Empirical Relevance

1. **Explaining Mode Collapse:** The authors show that mode collapse in VAEs and GANs can be understood as a consequence of manifold overfitting, where model densities diverge along subsets of the manifold without capturing the true data distribution.

2. **Diffusion Models:** The success of diffusion models is attributed to their ability to implicitly account for manifold structure by spreading probability mass beyond the manifold. The authors provide a detailed analysis of score-based diffusion models and their latent variants.

3. **Evaluation Metrics:** The paper advocates for the use of sample-based metrics, such as the Fréchet Inception Distance (FID), instead of likelihood-based metrics, which are shown to be unreliable in the manifold setting.

Conclusion

This paper provides a rigorous and insightful exploration of DGMs through the lens of the manifold hypothesis. By identifying the limitations of likelihood-based methods and highlighting the advantages of Wasserstein distances and two-step models, the authors pave the way for the development of more effective generative models. Their work not only advances theoretical understanding but also offers practical guidance for designing next-generation DGMs.

https://arxiv.org/abs/2404.02954

Mike's Daily Paper - 06.02.25
SMALL LANGUAGE MODELS: SURVEY, MEASUREMENTS, AND INSIGHTS

Introduction and Key Motivations:

This paper explores the growing significance of Small Language Models (SLMs) and compares their development to LLMs. While LLMs demand substantial computational resources and are typically deployed in data centers, SLMs are designed to function on resource-limited devices like laptops, tablets, smartphones and IoT devices. The research offers a comprehensive survey of 59 SLMs, evaluating them based on architectural advancements, training algorithms, and inference efficiency. By advocating for SLM adoption, this work aims to make machine intelligence more accessible, affordable, and efficient for practical deployment.

Key Contributions:

1. **Comprehensive Survey:**
   The authors provide a detailed landscape of 59 SLMs, identifying their unique architectural elements and performance benchmarks.
2. **Benchmarking and Evaluation:**
   The paper presents an in-depth evaluation of SLMs across several domains, including

commonsense reasoning, problem-solving, and mathematics, while analyzing inference latency, memory usage, and hardware performance.

3. **Insights for Model Optimization:**
   The study highlights trends and actionable insights for improving SLM efficiency, training strategies, and deployment.

Technical Analysis

1. Architectural Innovations

The research paper thoroughly explores the architectural elements that differentiate SLMs from LLMs, emphasizing the modifications that enhance efficiency on devices with limited resources.

- **Self-Attention Mechanisms:**
  Traditionally, Multi-Head Attention (MHA) was the dominant mechanism in SLMs. However, the paper observes a gradual shift toward Group-Query Attention (GQA) technique. This variant reduces computational complexity by sharing query representations across heads while maintaining diversity in key-value representations. The report provides evidence that GQA models, such as Qwen 2.5, significantly outperform those with MHA in terms of both latency and memory efficiency, particularly during the inference stage.

- **Feed-Forward Networks (FFNs):**
  The architectural evolution shows a preference for Gated FFNs over Standard FFNs. The Gated FFN introduces a gating mechanism that selectively activates parts of the network, leading to better parameter efficiency.
  An interesting finding is the diversification of intermediate ratios in Gated FFNs, ranging from 2 to 8 times the hidden dimension, with larger ratios generally improving accuracy on complex reasoning tasks.

- **Activation Functions:**
  A notable shift from GELU (Gaussian Error Linear Unit) and its variants to SiLU (Sigmoid Linear Unit) is observed. SiLU, being computationally efficient and better suited for low-precision arithmetic, dominates models released in 2024.

- **Layer Normalization:**
  The transition from LayerNorm to RMSNorm is highlighted. RMSNorm reduces the computational burden by eliminating the need to compute the mean during normalization, a valuable advantage for edge deployments.

- **Vocabulary Size:**
  The vocabulary sizes of SLMs have grown significantly, often exceeding 50,000 tokens. The authors associate this increase with improved language understanding capabilities and better handling of rare tokens, though it also contributes to larger memory footprints.

2. Training Dataset Analysis

The paper explores the datasets used to pretrain SLMs, revealing crucial insights into the evolving data landscape:

- **Dataset Usage Trends:**
  The study documents a shift from widely used general-purpose datasets such as *The Pile* and *RefinedWeb* to curated datasets like *FineWeb-Edu* and *DCLM*. These newer datasets incorporate model-based filtering techniques that significantly enhance data quality.

- **Data Quality vs. Quantity:**
  Despite earlier reliance on sheer data volume, the report finds that high-quality datasets yield better model performance, even with fewer tokens. For instance, models trained on *FineWeb-Edu* achieve competitive accuracy with state-of-the-art closed-source models.

- **Over-Training Observations:**
  The authors note a surprising trend: many SLMs are trained on token counts far exceeding what the Chinchilla Law suggests. For example, Qwen2.0 (500M parameters) is trained on 12 trillion tokens, while Qwen1.5 (1.5B parameters) is trained on only 7 trillion. This deliberate "over-training" strategy is posited as an optimization for resource-constrained environments, allowing models to generalize better when deployed on devices with limited computational power.

3. Training Algorithm Innovations

The paper examines several novel training algorithms that enhance SLM performance:

- **Maximal Update Parameterization (μP):**
  Used in models like Cerebras-GPT, μP ensures stable training by controlling initialization, layer-wise learning rates, and activation magnitudes. This technique allows hyperparameters optimized for small models to be directly transferred to larger models, streamlining the training process.

- **Knowledge Distillation:**
  LaMini-GPT and Gemma-2 leverage this technique to transfer knowledge from larger teacher models to smaller student models, resulting in improved performance without the need for extensive training.

- **Two-Stage Pretraining Strategy:**
  Adopted by MiniCPM, this strategy involves an initial phase with coarse-quality data followed by fine-tuning with high-quality, task-specific data. The method proves effective in balancing computational efficiency and model performance.

4. Performance Evaluation

The paper provides a rigorous assessment of SLM capabilities across multiple domains, including commonsense reasoning, problem-solving, and mathematics:

- **Commonsense Reasoning:**
  Models such as Phi-3-mini achieve state-of-the-art performance, rivaling LLaMA 3.1 (7B parameters). Benchmark results reveal that improvements in dataset quality and training strategies have allowed SLMs to close the gap with larger models.
- **Problem-Solving (with reasoning):**

- Phi-3-mini and other high-performing SLMs exhibit a 13.5% performance gain from 2022 to 2024, surpassing the improvement rate of LLaMA models. This demonstrates the increasing maturity of SLMs in handling complex reasoning tasks.

- **Mathematics:**
  SLM performance remains suboptimal in mathematics, with models struggling to handle tasks requiring logical reasoning. The authors attribute this gap to the lack of high-quality logic-focused datasets.

- **In-context Learning:**
  Experiments reveal that SLMs benefit significantly from in-context learning, particularly for tasks like ARC Challenge, where accuracy improvements of up to 4.8% are observed. However, some models, such as LaMini, exhibit performance degradation due to overfitting.

5. Runtime Efficiency Analysis

The authors perform extensive benchmarks to analyze inference latency, memory usage, and the impact of quantization and hardware:

- **Latency and Memory Footprint:**
  The study finds that inference latency is influenced by both model size and architecture. For instance, Qwen 1.5 (0.5B parameters) runs 31.9% faster than Qwen 2.0 (0.5B parameters) on Jetson Orin, despite having 25.4% more parameters. This is attributed to differences in attention mechanisms and parameter sharing strategies. Memory usage is generally linear with model size but is also affected by factors such as vocabulary size and attention mechanisms. Models like Bloom-1B, which have larger vocabularies, exhibit disproportionately high memory usage.

- **Quantization:**
  Quantization techniques, particularly 4-bit quantization, prove effective for reducing latency and memory usage. The Q4 K M method reduces latency by an average of 50% during inference, outperforming 3-bit and 6-bit quantization methods, which suffer from hardware inefficiencies.

Conclusion

The technical analysis presented in this paper provides a comprehensive understanding of the architectural, training, and runtime considerations essential for the development and deployment of SLMs. By addressing the challenges of efficiency and resource constraints, the paper offers valuable insights for advancing SLM research and practical applications.

https://arxiv.org/abs/2409.15790

Mike's Daily Paper - 08.02.25
Rejection Sampling IMLE: Designing Priors for Better Few-Shot Image Synthesis

Today we're taking a brief break from LLMs to review a paper that proposes an interesting method for training generative models when you have limited training data. As we know, modern generative models like diffusion models, GANs, and VAEs require enormous amounts of data, but sometimes we don't have this luxury and need to train on a small amount of data. Is this even possible?

The answer is positive (at least according to the paper). The authors propose a method called RS-IMLE for training a generative model with limited data, which improves upon the IMLE (Implicit Maximum Likelihood Estimation) method. Very broadly, IMLE is quite similar to a standard generative method - it samples a variable from an easy-to-sample distribution (Gaussian) and trains a generative model (neural network) to generate a piece of data. The difference is in the loss function: with IMLE, for each sample x from the dataset, we minimize only the distance between it and a single z_i point: one for which T(z_i) is closest to it. Here T(z_i) is a piece of data generated from z_i and T is the model we're training.

In other words, in the first stage of IMLE, we sample m points and pass them through model T (will be referred to as mapping) and generate m data points. Then for each sample x_j from the training dataset, we choose the z_i closest to x_j. Finally, only such z_i participate in minimizing the loss function. Obviously, the number of points m generated in the first stage needs to be significantly higher than the training dataset size n. The goal of this training method is to optimize the model only for points in the latent space (z) that are mapped close to points from the dataset.

The problem with this approach is that the distribution of the "selected" points during training is no longer Gaussian, which can be problematic during inference because we do want to sample z from a Gaussian distribution. The distance between mapping T of a normally distributed z sample from a dataset point differs in distribution from that of the z mapped closest to this point

(this is quite obvious). By the way, the paper proves this claim and proposes a method to overcome it.

The method proposed by the paper looks really simple but is based on quite deep mathematical analysis of distance distributions between T(z) and x. In the first stage of training (after sampling from Gaussian distribution), we pick z_i-s which fall at a distance greater than epsilon(hyperparameter) from all points in the training dataset after mapping T (i.e., we have rejection sampling here). Then, similar to IMLE, for each x in the dataset, we choose the z (out of the remaining samples) whose mapping with T falls closest to it and train T to minimize the average distance between the chosen z-s and their anchor points in the train dataset. The important hyperparameters here are epsilon and the number of sampled z points.

Intuitively, this works because we initially choose points that are farther away (after T) from the dataset points, which allows maintaining the distribution of selected points in the subsequent stage close to Gaussian.

https://arxiv.org/abs/2409.17439

Mike's Daily Paper - 09.02.25
Why Is Anything Conscious?

Introduction:

The paper *Why Is Anything Conscious?* by Michael Timothy Bennett, Sean Welsh, and Anna Ciaunica addresses the "hard problem of consciousness," famously articulated by David Chalmers. This philosophical challenge questions why information processing in certain systems, particularly biological ones, results in subjective experiences or *qualia*. The authors propose a paradigm shift, grounding consciousness in the dynamics of embodied, self-organizing systems shaped by natural selection.

They assert that phenomenal consciousness—the subjective "what it is like" to experience—is not only foundational but necessary for adaptive behavior. Through a formal computational framework, they argue against the possibility of "zombies," systems that function like humans but lack subjective experience, stating provocatively that "Nature does not like zombies."

Key Contributions

Mathematical Framework for Pancomputational Enactivism

The authors present a formal system rooted in *pancomputationalism* and *enactivism*. Pancomputationalism posits that all dynamic systems compute something, while enactivism emphasizes cognition as arising from interactions between a system and its environment. Central elements of their model include:

- Environment: Defined as a set of states, with transitions captured by declarative programs.

- Abstraction Layer: Structures how systems interpret environmental aspects.

- Tasks and Policies: Behavioral constructs that map inputs to outputs, facilitating adaptive behavior.

- Causal Identities: Representations of interventions and their effects, essential for self-awareness.

The framework describes how conscious systems maintain coherence and adaptability by constructing increasingly complex causal identities, forming the basis for self-awareness.

Hierarchy of Consciousness

A key insight is the hierarchical development of consciousness, driven by natural selection and scaling pressures. The authors outline six progressive stages:

1. Unconscious Systems: Entities devoid of experience or cognition, like rocks.

2. Hard-Coded Systems: Systems with fixed, preprogrammed responses (e.g., protozoa).

3. Learning Systems: Adaptable systems without self-awareness (e.g., nematodes).

4. First-Order Self Systems: Capable of distinguishing self-generated actions from external events (e.g., houseflies).

5. Second-Order Self Systems: Capable of meta-representation and intentional communication (e.g., ravens).

6. Third-Order Self Systems: Fully self-reflective beings capable of reasoning about their own awareness (e.g., humans).

This hierarchy underscores how qualitative aspects of consciousness naturally emerge as systems become more capable of modeling themselves and their environments.

Qualitative and Quantitative Processing

The authors argue that *quality precedes quantity* in information processing. Before an organism can label or measure information, it must experience qualitative differences. Phenomenal consciousness emerges because living systems must classify and prioritize

information relevant to survival. These qualitative classifications form the foundation for subjective experience.

This claim challenges traditional computational theories, which often treat consciousness as a purely representational process. By emphasizing the primacy of qualitative experience, the authors provide a fresh perspective on the origins of consciousness.

First Principles Approach

The formalism in the paper is derived from two basic axioms:

1. *Where there are things, we call these things the environment.*

2. *Where things differ, we have different states of the environment.*

These axioms lead to a representationless form of pancomputationalism, where states and transitions define environments without assuming specific internal structures. The authors frame self-organization as the capacity to constrain outputs based on inputs, achieving adaptive behavior.

Rejection of Zombies

One of the paper's boldest claims is that "Nature does not like zombies." The authors argue that phenomenal consciousness is essential for access consciousness and adaptive behavior. Representational content—what organisms reason about—is always derived from qualitative experience. Therefore, a system behaving like a conscious being must necessarily have subjective experience.

This claim directly challenges thought experiments that propose the existence of unconscious yet behaviorally indistinguishable entities.

Empirical Connections

The paper builds on empirical findings about *reafference*, or the ability to distinguish between self-generated and external stimuli. Reafference, observed in mammals and insects, is linked to the formation of a first-order self. The authors derive this construct from mathematical principles and align their conclusions with the work of Merker, Barron, and Klein.

Future Directions

Empirical Testing:

The hierarchical framework invites experimental validation. Researchers could investigate the neural correlates of first-order and second-order selves in animals known for complex social behavior. Understanding how organisms construct causal identities would be a promising avenue for further study.

Applications to Artificial Intelligence

The formalism has significant implications for AI research. By framing consciousness as an emergent property of self-organizing systems, the authors suggest new pathways for creating adaptive, self-aware artificial agents.

Ethical and Philosophical Implications

The rejection of zombies implies that many non-human animals may possess forms of consciousness deserving ethical consideration. Additionally, the framework raises questions about the moral status of artificial systems capable of subjective experience.

Conclusion

Why Is Anything Conscious? offers a groundbreaking approach to the hard problem of consciousness by grounding it in natural selection, self-organization, and computational formalism. The authors' hierarchical framework provides a compelling explanation for how consciousness emerges and why subjective experience is fundamental to adaptive behavior. Their provocative claim that zombies are impossible challenges long-standing assumptions, marking this paper as a significant contribution to consciousness studies.

Mike's Daily Paper - 10.02.25
On the expressiveness and spectral bias of KANs.

This paper delivers a thorough investigation of Kolmogorov-Arnold Networks (KANs), a novel architecture inspired by the Kolmogorov-Arnold representation theorem. The authors rigorously compare KANs to traditional multi-layer perceptrons (MLPs) both theoretically and empirically, focusing on aspects like expressiveness, efficiency, and training dynamics. The paper breaks new ground by establishing key theoretical properties and validating them through experiments, making it a significant contribution to neural network design for scientific computing tasks.

Expressiveness and Approximation Power

A primary achievement of this work is the formal demonstration that KANs are at least as expressive as MLPs. The authors show that any ReLU-based MLP can be restructured into a comparable KAN architecture, maintaining efficiency without a substantial increase in network size. Conversely, while KANs can also be represented by MLPs, this transformation incurs a significant cost: the number of parameters scales with the grid size of the KAN. This suggests that KANs may offer more efficient representations for certain classes of functions, particularly when fine grid structures are used.

The study further leverages existing results for MLPs to establish approximation rates for KANs in function spaces like Sobolev spaces. It demonstrates that KANs achieve comparable or better approximation rates than MLPs in capturing complex function structures, reinforcing their theoretical robustness.

Spectral Bias Analysis

One of the key distinctions between KANs and MLPs highlighted in this paper is the difference in their spectral bias — a phenomenon where neural networks tend to learn low-frequency components of functions first. The authors present a detailed theoretical and empirical analysis, showing that KANs suffer significantly less from this bias.

This difference is attributed to the B-spline-based activation functions and the compositional architecture of KANs, which allow them to learn high-frequency components more efficiently. Theoretical insights suggest that the training dynamics of shallow KANs are more uniform across different frequencies compared to MLPs, whose convergence strongly favors lower frequencies. This reduced spectral bias makes KANs better suited for tasks requiring precise handling of high-frequency information, such as solving differential equations and modeling complex physical phenomena.

Empirical Validation

The authors provide extensive empirical evidence to validate their theoretical findings:

1. **Frequency Regression Tests:** KANs successfully fit high-frequency wave components simultaneously, while MLPs exhibit persistent struggles with higher frequencies even after prolonged training.
2. **Gaussian Random Field Experiments:** KANs outperform MLPs in approximating functions sampled from rough Gaussian fields, indicating superior adaptability to complex function structures.
3. **PDE Solutions:** In solving high-frequency Poisson equations, KANs achieve consistently lower errors compared to MLPs, maintaining stable performance even as the frequency of the solution increases.

These results illustrate the practical benefits of KANs for scientific computing tasks, where capturing detailed, high-frequency information is often crucial.

Grid Extension Technique

A notable technical innovation discussed in the paper is the grid extension technique specific to KANs. This method allows for progressive refinement of the spline grid during training, enabling a more efficient learning process. The grid extension approach mitigates the risks of overfitting and enhances the network's ability to generalize, particularly when dealing with complex functions or undersampled datasets.

Conclusion

This work establishes KANs as a powerful and efficient alternative to MLPs, particularly for tasks in scientific computing. By addressing spectral bias, enhancing approximation capabilities, and leveraging adaptive training methods, the authors provide compelling evidence for the potential of KANs to outperform traditional neural networks in applications requiring high-frequency

learning and precise function approximation. The rigorous theoretical framework combined with comprehensive experiments makes this paper a foundational contribution to neural network research.

https://arxiv.org/abs/2410.01803

Mike's Daily Paper - 11.02.25
STUFFED MAMBA: State Collapse and State Capacity of RNN-Based Long-Context Modeling

The paper provides a rigorous investigation into the failure modes of RNN-based models in long-context language modeling and proposes solutions to enhance their length generalization capabilities. The authors identify and analyze a crucial phenomenon termed State Collapse (SC) - a failure in recurrent state dynamics that prevents RNNs from extrapolating beyond their training lengths. They introduce a set of training-free mitigation techniques and continuation training strategies that enable their proposed Mamba-2 model to scale to over 1M tokens without suffering from SC.

1. **Problem Statement and Context**

1.1. RNNs vs. Transformers for Long-Context Modeling

- Transformers achieve superior performance in long-context reasoning but suffer from quadratic computational complexity in sequence length due to the attention mechanism.
- RNNs, in contrast, exhibit linear computational complexity with respect to sequence length, making them computationally efficient for handling long sequences.
- However, state-of-the-art RNN models (Mamba-1, RWKV, etc.) are trained on relatively short sequences (~10K tokens) and fail to generalize beyond training lengths.

1.2. Core Issues in Long-Context RNNs

The study identifies two fundamental problems:

1. **Failure to Extrapolate to Longer Sequences**
    - RNNs exhibit a sharp degradation in performance when exposed to sequence lengths beyond their training data.
    - This failure is not simply due to vanishing gradients but is attributed to state collapse (SC).
2. Fixed Recurrent State Memory Capacity
    - Since RNNs maintain a constant-size memory state, their ability to retain information is inherently bounded.
    - There exists an upper limit on the contextual memory capacity—tokens beyond this limit are inevitably forgotten.

2. Formal Analysis of State Collapse (SC)

2.1. Definition and Experimental Evidence

- State Collapse (SC) occurs when the hidden recurrent state distribution collapses, leading to model failure in processing sequences longer than the training set.
- The authors conduct controlled experiments on Mamba-2 and observe that certain hidden state channels exhibit variance explosion, which causes:
    - Dominant outlier channels that suppress other state values.
    - Inability to forget earlier tokens, leading to an overflow of memory.
- SC manifests as a sharp increase in perplexity beyond training length.

2.2. Theoretical Attribution: Overparameterization in State Dynamics

The authors derive the state update equation:

$$h_t = \sum_{i=1}^{t} \alpha_{i:t} \overline{B}_i x_i, \quad \alpha_{i:t} = \left(\prod_{j=i}^{t} \alpha_j \right) \in (0, 1)$$

where:

- h_t is the hidden recurrent state.
- $\alpha_{i:t}$ represents the memory decay factor.
- B_i x_i represents new information inserted at time step i.

Key observation:

When training on sequences of length T_train, the learned model parameters favor retaining all information within T_train, failing to forget when processing longer sequences. This results in over-accumulation of information, which leads to state saturation and eventual collapse.

3. Proposed Mitigation Strategies

3.1. Training-Free SC Mitigation Techniques

The authors propose three training-free methods to suppress SC:

Controlled Forgetting:

- Increase state decay by modifying the decay factor α_t.
- Reduce insertion strength (i.e., B_t).
- Effectively forces the model to forget older tokens, preventing state overflow.

State Normalization:

- Apply a norm-based constraint to the recurrent state:

$$\hat{h}_t = h_{t-1}\overline{A}_t + \overline{B}^T x_t$$
$$h_t = \begin{cases} \hat{h}_t p / \|\hat{h}_t\| & \text{if } \|\hat{h}_t\| > p \\ \hat{h}_t & \text{if } \|\hat{h}_t\| \leq p \end{cases}$$

where p is a predefined threshold. It prevents hidden state explosions but introduces non-linearity, which affects training efficiency.

Sliding Window State Update

- Reformulate the state update rule to simulate a sliding window mechanism:

$$h_t^{(r)} = \sum_{i=t-r+1}^{t} \alpha_{i:t}\hat{R}_i = \sum_{i=1}^{t} \alpha_{i:t}\overline{B}_i^T x_i - \alpha_{t-r+1:t}\sum_{i=1}^{t-r} \alpha_{i:t-r}\overline{B}_i^T x_i = h_t - \alpha_{t-r+1:t}h_{t-r}$$

- This effectively removes older tokens without recomputing from scratch. It is applicable to other RNN architectures like RWKV and RetNet.

3.2. Continued Training on Longer Sequences

- The authors extend training lengths beyond state capacity to force the model to learn how to gradually forget.
- They empirically verify that for any state size SS, there exists a threshold training length T_train where SC does not occur.

4. Conclusion

4.1. Major Contributions

- First systematic study of state collapse in long-context RNNs.
- Proposed three training-free mitigation methods to eliminate SC up to 1M tokens.
- Empirically established the relationship between state size and capacity.
- Trained a 370M Mamba-2 model with 256K-token perfect retrieval—unprecedented at this scale.

4.2. Implications

- The findings challenge conventional training lengths used in RNN-based language models.
- The proposed state-aware modifications offer a viable alternative to transformers for long-context NLP.

6. Final Remarks

The paper combines rigorous theoretical analysis, empirical validation, and novel algorithmic improvements to address one of the fundamental limitations of RNN-based long-context models. By resolving state collapse, this work revives interest in RNN architectures for scalable, efficient long-context reasoning.

Future work could explore:

- Applying SC mitigation to other architectures (RWKV, RetNet, etc.).
- Enhancing adaptive state scaling to dynamically allocate memory capacity.
- Investigating prompt dependency in SC for better real-world generalization.

This work lays the foundation for efficient long-context modeling beyond transformer architectures, marking a significant step in the evolution of scalable sequence models.

Mike's Daily Paper - 13.02.25
One Initialization to Rule them All: Fine-tuning via Explained Variance Adaptation

Today we will briefly review a paper proposing a LoRA method for improving the fine-tuning technique of LLMs. As you probably remember, LoRA adds a learnable matrix with a significantly lower rank than the weight matrix dimension to the model weights (in certain layers). The model weights remain fixed (not trained) during fine-tuning.

The authors propose an approach to enhance LoRA that includes a preliminary stage called Data-Driven initialization in the paper. The purpose of this initialization is "to adjust the rank of LoRA matrices for each layer of the model." Since we're training certain weight additions, we can distribute them "optimally" between model layers. The optimality here is measured through the variance of the layer activations (i.e., the output of the FFN layer) for the data we're training on.

If the activation variance on training data is low, it means the layer values are more or less constant, and it's not worth wasting LoRA weights on it. In other words, we can use a very low rank LoRA (if at all) for this layer. But how can we measure this variance using singular values of activation matrices computed through SVD decomposition of this matrix? The dimensions of the activation matrix here are the hidden dimension of the model and the batch size.

So, we compute the singular values of the activation matrix on the training dataset until the singular vectors (right vectors) stabilize. These vectors are updated during batch runs (the model is trained on the fine-tune dataset) and their creation (of the vectors) stops when they stabilize and stop changing significantly (the paper measures similarity using cosine distance - if it's too high for a particular layer, vector updates for that layer stop) - this training is performed before LoRA.

After the singular vectors converge for all layers, we take the eigenvalues and calculate the percentage of variance explained by each layer (calculated by the sum of squares of their singular values) relative to the variance explained by the entire model (which is the sum of squares of singular values for activations of all model layers).

In the next stage, LoRA matrix ranks are allocated to layers as functions of their explained variance. That is, as the explained variance of a layer increases, more “LoRA ranks” are allocated. In the final stage, LoRA is trained with an "optimal" allocation of LoRA matrix ranks based on the training data. Quite an interesting idea that shows decent results.

https://arxiv.org/abs/2410.07170

Mike’s Daily Paper - 15.02.25
A Spectral Condition for Feature Learning

1. Introduction

The paper *"A Spectral Condition for Feature Learning"* by Greg Yang, James B. Simon, and Jeremy Bernstein presents a rigorous theoretical framework for understanding feature learning in deep neural networks through the lens of spectral norm scaling. The authors introduce a spectral scaling condition that governs the evolution of network features during training, providing a refined alternative to heuristic-based initialization and learning rate scaling strategies.

The primary motivation of this work is to address a key challenge in scaling large-width neural networks: ensuring that feature learning occurs effectively across all layers, preventing both gradient vanishing and feature explosion. The authors propose that by appropriately scaling the spectral norm of weight matrices and their updates, feature learning can be preserved even in the infinite-width limit. This framework offers a more principled approach compared to traditional Frobenius norm-based initialization schemes.

The paper contributes to both theoretical and practical aspects of deep learning training by demonstrating how spectral norm considerations naturally lead to the *Maximal Update Parametrization (μP)*, an initialization and learning rate scaling strategy that allows hyperparameter transfer from narrow to wide models. Unlike previous works that derived μP using tensor program arguments, this paper provides an elementary linear algebra-based derivation, making it more accessible to the broader deep learning community.

2. Core Contributions and Theoretical Foundations

2.1 The Spectral Scaling Condition

The central result of the paper is a scaling condition on the spectral norm of weight matrices and their gradient updates:

$$||W_{l}\|_* = \Theta\left(\sqrt{\frac{n_{l}}{n_{l-1}}}\right), \quad \|\Delta W_{l}\|_* = \Theta\left(\sqrt{\frac{n_{l}}{n_{l-1}}}\right)$$

where n_l and n_{l-1} are the fan-out and fan-in dimensions at layer l. This condition ensures that both the hidden features h_l and their updates Δh_l remain at an appropriate scale:

$$\large{\|h_{l}\|_2} = \Theta(\sqrt{n_{l}}), \quad \|\Delta h_{l}\|_2 = \Theta(\sqrt{n_{l}})$$

which prevents feature explosion or vanishing and enables stable learning dynamics across layers. The motivation behind this condition stems from the way information propagates through neural networks. In traditional initialization schemes, such as Kaiming or Xavier, the Frobenius norm is used as a proxy for controlling the scale of activations. However, the authors argue that the spectral norm - a measure of the largest singular value of a matrix—is a more appropriate metric for controlling the effective amplification of signals across layers.

2.2 Justification via Matrix Analysis and Feature Learning

The spectral scaling condition arises from a fundamental property of deep networks: each layer applies a transformation that amplifies or attenuates input signals according to the singular values of the weight matrix. The largest singular value (which defines the spectral norm) determines how much a layer can stretch or shrink input activations along specific directions in feature space.

By ensuring that the spectral norm follows the prescribed scaling, the authors prove that:

- Feature magnitudes remain stable across layers, neither vanishing nor exploding.
- The evolution of features during training remains significant, preventing a collapse into trivial representations.

To rigorously justify this, the paper provides an in-depth mathematical analysis of gradient updates in multilayer perceptrons (MLPs). A key insight is that weight updates in deep networks exhibit a rank-one structure due to the outer-product nature of gradients:

$$\Delta W_{ll} = -\eta_{l} \nabla_{W_{l}} L = -\eta_{l} \cdot (\text{error signal}) \cdot (\text{input features})^T$$

This structure leads to the observation that weight updates naturally align with dominant singular vectors of weight matrices, reinforcing the spectral norm as the primary determinant of network evolution.

The paper shows that under this spectral scaling condition, networks maintain meaningful feature learning at all widths. In contrast, standard parameterization (SP) and neural tangent parameterization (NTP) fail to maintain feature evolution in the infinite-width limit.

2.3 Relation to Maximal Update Parametrization (μP)

One of the most impactful contributions of the paper is its connection to *Maximal Update Parametrization (μP)*. μP, introduced by Yang & Hu (2021), prescribes an initialization and learning rate scaling that allows hyperparameters to transfer from small to large models without additional tuning. Previously, μP was derived through tensor program arguments, which are mathematically intricate.

This paper provides a much simpler derivation using spectral norm scaling. The authors show that μP is equivalent to ensuring that weight matrices and their updates obey the spectral scaling condition:

$$\sigma_{l} = \Theta\left(\frac{1}{\sqrt{n_{l-1}}} \min\left\{1, \sqrt{\frac{n_{l}}{n_{l-1}}}\right\} \right), \quad \eta_{l} = \Theta\left(\frac{n_{l}}{n_{l-1}}\right)$$

where σ_l *is the weight initialization scale and* η_l is the learning rate at layer l. This formulation generalizes μP to arbitrary layer shapes, eliminating the need for special-case rules for input, hidden, and output layers. Moreover, it clarifies why μP preserves feature learning: the spectral norm scaling ensures that both weight updates and feature updates remain order-one, preventing collapse into the kernel regime seen in NTP.

2.4 Empirical Validation and Comparisons

The authors provide empirical support for their theoretical claims by analyzing the behavior of MLPs trained on CIFAR-10. The key observations include:

1. **Low-Rank Structure of Weight Updates**
    - Even at large batch sizes, gradient updates remain low-rank, meaning that weight updates are effectively controlled by their spectral norm rather than their Frobenius norm.
2. **Feature Evolution Across Training**
    - Feature representations evolve significantly under μP but decay under NTP, confirming that NTP fails to achieve proper feature learning.
    - The evolution of features follows the predicted Θ(1) scaling under spectral normalization but collapses under traditional parameterizations.
3. **Comparing Spectral vs. Frobenius Scaling**
    - The study highlights that Frobenius norm scaling leads to underestimated weight updates due to low-rank gradient structures.
    - The spectral condition correctly preserves feature evolution even in deep networks.

These experimental results strongly reinforce the theoretical arguments made in the paper, demonstrating that spectral scaling is not merely a theoretical construct but has direct practical implications for neural network training.

https://arxiv.org/abs/2310.17813

Mike's Daily Article: 16.02.25
Representation Alignment for Generation: Training Diffusion Transformers is Easier than You Think

Taking a brief break from LLMs to review a paper about generative diffusion models. The paper proposes a rather intuitive method for improving the performance of these models by adding a regularization term that "aligns" the model's internal representations with those of powerful encoders like DiNOV2. This alignment improves the quality of images generated by the model.

Let's start with a brief background on generative diffusion models. These models are trained to generate images (for example, given a textual description) through gradual noise removal. The model starts with pure noise (typically Gaussian) and slowly transforms it into an image (or data from another domain). The model is trained on noisy images with different noise levels (=iterations), where during training the model learns to remove a small amount of noise (from iteration t to iteration t-1). The choice of hyperparameter t for the noise process is critical to the generation quality of the trained model.

This process (noise addition) can be described using differential equations of probability flow that describe the change (gradient) of the noisy data with noise rate/velocity that we'll denote as vt (the solution to this equation is distributed according to the noisy data distribution). The noise rate can be estimated with the model (=network) based on noisy data samples and t. Then, the probability flow equations can be solved with the estimate of vt (in reverse direction - i.e., starting from pure noise) using Euler's method, for example. These methods are called stochastic interpolands. Note that there are methods based on numerical solutions of stochastic differential equations that describe data changes as a function of the score function, which is the logarithm of the noisy data distribution function.

Okay, after this complexity things get a bit easier. Diffusion models today are mostly latent models where generation occurs in the data representation space. This means the model is trained to recover a latent representation from noise, and then the decoder is applied to construct an image from the recovered representation. The representation of the initial image is created by the encoder. The authors argue that the noisy latent representations aren't "strong enough," meaning they less effectively reflect the semantic aspects of the image.

The authors propose to enrich these representations by adding a regularization term aimed at bringing these representations (of noisy images) closer to the representation produced by a strong encoder (like DINOV2). This loss is added to the regular diffusion model loss, and it's claimed to improve the quality of generated images and contribute to training stability.

https://arxiv.org/abs/2410.06940

Mike's Daily Paper - 18.02.25
THINKING LLMS: GENERAL INSTRUCTION FOLLOWING WITH THOUGHT GENERATION

Deep Learning Paper Review Number 400: To avoid overwhelming you, I chose a relatively light paper, and the review will be without formulas and quite brief. The paper proposes a method somewhat similar in essence to Group Relative Preference Optimization, or GRPO for short, which has made many headlines recently. I'll explain what I mean by this shortly. I'll just

note that the paper proposes a method to enhance general reasoning capability of a model and doesn't focus only on programming questions and mathematical problems.

The paper proposes a fine-tuning method for LLMs that enhances their reasoning capabilities without requiring labeled data. The paper suggests training in an RLHF-style, but unlike DeepSeek (the inventor of GRPO), the authors proposed using the DPO method that doesn't use a reward function at all. I'll note that GRPO doesn't train a reward model like PPO does, but rather uses the correctness of the answer and its format as a reward function.

So what's common between GRPO and the method proposed in this paper? Both essentially suggest not penalizing the model for its “thinking process”/chain of thought (which might be incorrect but can lead to the correct answer), but judging it only based on the correctness of the model's answer (as mentioned, GRPO also penalizes for not adhering to the answer format). After we understand the fundamental connections between the proposed method and the famous methods, let's dive into what the paper proposes.

As mentioned, the paper proposes to fine-tune a language model's reasoning ability without using labeled data with RLHF. As you remember, RLHF with DPO requires pairs of preferred and less preferred answers. Since we said the method doesn't require labeled data, you can guess that building these pairs is done by a judge language model that selects good and bad answers similar to the RLAIF method, which stands for Reinforcement Learning from AI Feedback. The judge model operates on answers (not the reasoning chain!) of the trained model and decides which among the answers is the best and worst. These pairs are used to train the model in DPO form. Of course, there's also engineering of a meta-prompt causing the model to "generate a thinking process" but the reasoning chain, generated by it, doesn't participate in training the model. Namely the reward is computed solely based on the answer correctness.

https://arxiv.org/abs/2410.10630

Mike’s Daily Paper - 19.02.25
Losing Dimensions: Geometric Memorization in Generative Diffusion Models

The paper introduces a novel theoretical framework for understanding how diffusion models memorize and generalize data, particularly when the data is supported on a low-dimensional manifold. It extends prior work on memorization in generative diffusion models by incorporating a geometric perspective, providing new insights into how different tangent subspaces of the data manifold are selectively lost due to memorization effects. The study leverages statistical physics, random matrix theory, and differential geometry to develop a comprehensive explanation of geometric memorization.

Core Novelty:

1. **Geometric Perspective on Memorization:**
   The paper presents a novel characterization of memorization in generative diffusion

models as a process that leads to a **selective loss of dimensionality** rather than an outright collapse onto individual training points.
    - It establishes that different tangent subspaces of the data manifold are lost at different critical times, depending on the local variance of the data along those directions.
    - This structured degradation contrasts with classical views of memorization, which often assume uniform or complete collapse of the data manifold.

2. **Variance-Dependent Dimensionality Loss:**
    - The study reveals an **unexpected phenomenon**: **subspaces with higher variance are lost first due to memorization effects**. This contradicts intuition, as one would expect that lower-variance, less informative dimensions would be lost first.
    - The result is a hierarchical loss of generative capacity, where high-variance directions disappear before low-variance ones, leading to a structured, non-trivial form of memorization.

3. **Random Matrix Theory and Spectral Gap Analysis:**
    - The authors employ techniques from **random matrix theory** to analyze the spectral properties of the Jacobian of the score function in diffusion models.
    - They demonstrate that the **closure of spectral gaps** in the singular values of this Jacobian corresponds to the loss of generative flexibility along specific directions.
    - The study quantifies how the number of effective generative degrees of freedom evolves as a function of dataset size and diffusion time.

4. **Statistical Physics Interpretation and Phase Transitions:**
    - The paper frames memorization as a **glassy phase transition**, drawing parallels to associative memory networks and statistical mechanics models.
    - The transition from generalization to memorization is shown to follow a condensation process similar to those observed in disordered physical systems.
    - Using the **Random Energy Model (REM) analogy**, the study provides an analytical estimate of the critical diffusion time at which memorization effects dominate.

5. **Empirical Validation Across Manifold-Supported and Natural Image Data:**
    - The authors validate their theory through controlled experiments on synthetic data (e.g., linear manifolds with varying variance subspaces) and real-world datasets (MNIST, CIFAR-10, CelebA).
    - The experimental results **strongly align with the theoretical predictions**, demonstrating that trained diffusion models exhibit the predicted **hierarchical loss of dimensionality** as dataset size decreases.
    - The paper introduces a novel method for estimating the **intrinsic dimensionality of generated samples**, which corroborates the theoretical framework.

https://arxiv.org/abs/2410.08727

Mike's Daily Paper – 22.02.25
When Does Perceptual Alignment Benefit Vision Representations?

Taking a brief break from language models, today we review a paper in the field of computer vision. The authors propose a method to improve visual data embeddings by fine-tuning the embedding model (or backbone) on a dataset of images labeled as similar or dissimilar by humans. Empirical results show that this fine-tuning enhances the quality of learned representations for various vision tasks such as object counting, segmentation, depth estimation, and instance retrieval.

The fine-tuning utilizes **contrastive learning**, a method I've covered extensively, reviewing dozens of papers on this fascinating topic. In general, contrastive learning aims to bring embeddings of similar data points closer together (typically measured using cosine distance, though other options exist) while pushing apart embeddings of dissimilar data points. The authors use the **NIGHTS dataset**, which contains triplets of images with roughly the same semantic content but differing in pose, shape, color, and object count.

The authors adopt a **triplet loss** approach to contrastive learning, training the model on image triplets. Each triplet consists of an anchor image (reference) and two additional images labeled by human annotators regarding their similarity to the anchor. Annotators were asked to indicate which of the two images is more or less similar to the anchor. The training objective is to maximize the difference between the cosine distance of the more similar image to the anchor and that of the less similar image.

Additionally, the authors propose contrastive learning on the concatenation of the image embedding with the average representation of all its patches. They claim this improves results to some extent.

Paper link: https://arxiv.org/abs/2410.10817

Mike's Daily Paper – 22.02.25
Addition Is All You Need: For Energy-Efficient Language Models

The paper presents an elegant yet radical approach to improving the efficiency of neural networks, particularly in the context of LLMs. The authors propose an alternative to traditional floating-point multiplications, called Linear-Complexity Multiplication (L-Mul), which approximates floating-point operations using integer additions. The central claim is that L-Mul significantly reduces computational complexity and energy consumption while maintaining nearly identical model performance.

Motivation:
The motivation is clear: the energy demands of AI systems, especially large models, are

becoming unsustainable. Floating-point multiplications are among the most computationally expensive operations, and replacing them with more efficient alternatives could have significant implications for hardware design and practical AI applications. The authors highlight how energy consumption in neural networks increases with the number of floating-point operations and quantify the potential energy savings by replacing multiplications with additions.

Technical Explanation:
Traditional floating-point multiplication involves costly exponent and mantissa operations. L-Mul bypasses this by restructuring the computation, using integer addition instead of mantissa multiplication. The authors support this with a theoretical error estimation, demonstrating that L-Mul with a 3-bit mantissa outperforms float8 e5m2 multiplication, while a 4-bit mantissa competes favorably with float8 e4m3. This mathematical rigor provides strong credibility to their claims.

Experimental validation:
The authors integrate L-Mul into transformer-based language models and assess its effectiveness across various tasks, including natural language understanding, commonsense reasoning, and mathematical problem-solving. The results are promising: applying L-Mul to the attention mechanism results in negligible accuracy loss and, in some cases, minor performance improvements. The authors even show that fully replacing all floating-point multiplications with a 3-bit mantissa L-Mul in a transformer model yields results similar to float8 e4m3 in both fine-tuning and inference.

Pros and Cons:
One of the most compelling aspects of the paper is its focus on energy efficiency. Using data from previous studies on hardware energy consumption, the authors estimate that L-Mul can reduce the energy cost of element-wise multiplications by 95% and dot product operations by 80%. This is a bold claim, suggesting that L-Mul could have immediate and tangible benefits for data centers and large-scale AI applications.

Despite its advantages, the paper leaves some practical questions unanswered. The authors acknowledge that existing GPUs lack native support for L-Mul, making efficient implementation challenging. While they suggest that specialized hardware could optimize L-Mul-based computations, they do not provide a concrete roadmap for hardware adoption. Additionally, while the theoretical and empirical precision analyses are compelling, real-world deployment would require extensive testing in production environments.

Conclusion:
The paper presents an innovative and well-supported approach to reducing the computational and energy costs of large language models. The theoretical grounding is strong, the experimental results are convincing, and the potential impact is significant. While practical challenges remain—especially in hardware adoption—this work opens new doors for

energy-efficient AI computation. If further refined and adopted, L-Mul could play a crucial role in making AI more sustainable without sacrificing performance.

https://arxiv.org/abs/2410.00907

Mike's Daily Paper – 25.02.25
Understanding Visual Feature Reliance through the Lens of Complexity

Introduction:

The paper I'm reviewing today presents an unusual, rare and interesting study on feature complexity in deep learning models (no RAG, agents and LLMs are in there :). And it is tightly connected to the idea of the information bottleneck in deep neural networks, coined by N.Tishbi.

This paper introduces a new information-theoretic framework for quantifying feature complexity in deep learning models, offering a mathematically principled approach to understanding how, when, and where features emerge during training. Unlike traditional interpretability methods that focus on saliency and attribution, the study presents v-information as a computational complexity-aware measure, capturing the effort required to extract a feature rather than just its direct statistical dependence on input data.

Through extensive empirical analysis, the paper systematically explores the temporal evolution, spatial distribution, and functional role of features in deep vision models. The findings suggest that deep neural nets exhibit a hierarchical learning process, where "simpler", lower-complexity features emerge early in training and propagate efficiently through residual connections, while more complex features require deeper processing(more complex computations) and extended training time but contribute less critically to final model decisions than previously assumed.

1. A Complexity-Aware View of Feature Learning

Feature analysis in deep learning has largely been framed in terms of importance and utility, but little work has been done to explicitly quantify how complex a feature is to extract from raw inputs (namely amount of computation required to compute it). This study shifts the focus from conventional feature relevance metrics to a measure of computational effort, formalized as v-information.

1.1 Redefine Feature Complexity with v-Information

Traditional feature evaluation approaches rely on direct mutual information estimates between feature activations and input data. However, this approach fails to account for the transformation difficulty required to obtain a feature.

The key innovation in this paper is the introduction of v-information, which quantifies:

- How much processing is required to extract a feature within a model.

- The depth and nonlinearity of transformations involved in computing that feature from raw inputs.
- The computational complexity of mapping input signals to the feature space, rather than simply measuring its statistical correlation with inputs.

1.2 Why Complexity Matters in Feature Learning

Prior studies in information bottleneck theory(extensively studied by N. Tishbi) suggest that deep models progressively refine input representations, discarding irrelevant information while preserving task-relevant signals. This work builds on these principles by providing a rigorous measure for determining which features require deeper transformations and which emerge naturally through shallower layers.

Empirical evidence supports the claim that models prioritize lower-complexity features early in training, while more complex features emerge gradually over longer training horizons. This aligns with existing implicit curriculum learning theories, which suggest that optimization landscapes favor learning low-complexity patterns first before refining higher-complexity abstractions.

1.3 Temporal Dynamics of Feature Complexity

One of the most compelling findings in the paper is that feature learning follows a structured progression over time:

- Early training: The model rapidly learns low-complexity features, which require fewer nonlinear transformations to extract.
- Mid-training: Intermediate features emerge, often composed of combinations of lower-complexity features.
- Late training: High-complexity features emerge, but their impact on final predictions is minimal compared to early-stage features.

This structured evolution suggests that the optimization process in deep networks naturally orders feature learning in a way that minimizes computational burden early on, allowing the model to efficiently bootstrap its representations before refining complex details.

1.4 Spatial Organization of Feature Complexity in Network Architectures

The study finds that feature complexity is not evenly distributed across layers but instead follows a structured organization:

- Lower-complexity features are found in earlier layers and can propagate via residual connections.
- Higher-complexity features require deeper processing and accumulate nonlinearly through stacked transformations.
- Residual pathways act as computational shortcuts for low-complexity features, allowing them to bypass deeper processing.

This finding suggests a new interpretation of residual networks (aka ResNets): rather than simply aiding gradient flow, residual connections serve as complexity filters, allowing lower-complexity features to be processed more efficiently.

1.5 Feature Complexity and Model Decision-Making

A crucial implication of this framework is that high-complexity features contribute less to final classification decisions than expected. Empirical evidence suggests that:

- Models rely more heavily on lower-complexity features for generalization.
- Complex features, while present, have a weaker correlation with final classification outputs.
- Overemphasis on complex features does not necessarily improve performance and may lead to overfitting.

This contradicts the traditional assumption that deep models primarily rely on highly abstract representations for decision-making. Instead, the study suggests that models exploit simple, robust signals whenever possible, leveraging complex features only as secondary refinements.

2. Experiments

The authors conduct a large-scale empirical study on feature complexity in deep vision models, analyzing 10K feature directions extracted from the penultimate layer of an ImageNet-trained ResNet-50 model. The experiments are designed to rigorously validate the relationship between feature complexity, temporal learning dynamics, and spatial distribution within the network.The authors conducted an extensive empirical study on feature complexity within deep vision models, analyzing 10,000 feature directions extracted from the penultimate layer of an ImageNet-trained ResNet-50 model. The experiments were designed to rigorously validate the relationship between feature complexity, temporal learning dynamics, and spatial distribution within the network.

2.1 Feature Extraction and Complexity Measurement

The authors extract feature directions from intermediate network activations and estimate their v-information complexity. This is achieved by:

- Isolating feature directions in the penultimate layer representation space.
- Computing how much processing is required to extract each feature.
- Ranking features based on their computational effort.

This approach allows the authors to build a hierarchical feature complexity map that quantifies the relative difficulty of learning each feature direction within the model.

2.2 Tracking Feature Complexity Over Training Time

To understand how and when features emerge, the authors analyze model checkpoints at different training stages. This enables them to systematically track:

- Which features appear early, mid, and late in training.
- How feature complexity correlates with learning speed.
- The stability of feature reliance throughout training.

2.3 Layer-Wise Complexity Distribution and Residual Path Analysis

A key experiment focuses on how feature complexity propagates across different layers. The authors find that:

- Simple features propagate effectively through shortcut paths in ResNets.
- Complex features require deep hierarchical processing.
- The presence of residual connections reduces the need for deep feature transformations for low-complexity features.

These findings suggest that network architectures inherently allocate resources to feature complexity processing, shaping how models encode information at different depths.

3. Conclusion:

This paper makes a significant theoretical and empirical contribution by introducing a mathematically grounded framework for analyzing feature complexity in deep learning models. The key insights include:

1. Feature learning follows a structured, curriculum-like progression, where simpler features emerge first.
2. Residual connections serve as computational shortcuts, allowing low-complexity features to bypass deep transformations.
3. Deep models rely more heavily on simple, robust features for decision-making, contradicting the assumption that high-complexity abstractions drive model predictions.
4. V-information provides a principled way to quantify feature complexity, offering a new tool for studying inductive biases in deep networks.

https://arxiv.org/abs/2407.06076

Mike's Daily Paper – 27.02.25
Unity by Diversity: Improved Representation Learning for Multimodal VAEs

Today, I'm returning to a paper on Variational Autoencoder (VAE) after a very long time—more than a year, I suppose.

To recap, a VAE is a type of generative model that learns to compress data into a structured low-dimensional latent space (with an induced distribution) and then reconstruct it. The main challenge is ensuring that this space remains "smooth", so that sampling from it produces

realistic data. To achieve this, a VAE balances two objectives: accurately reconstructing the original data while ensuring that the latent space stays close (in distribution) to a simple and well-defined prior, usually Gaussian. This guarantees that nearby points in the latent space correspond to similar data samples, making it possible to generate new, coherent examples.

Multimodal VAEs extend this idea to handle multiple types of data, such as images, text, and audio, within a unified framework. The challenge with MVAE arises from the fact that different modes (modalities) share some information but also contain unique details. Some MVAE models attempt to force all modalities to share a single latent representation, which can lead to the loss of modality-specific information. Others keep them too separate, preventing meaningful cross-modal interactions. The ideal MVAE must strike a balance: capturing the shared structure of different modes while preserving what makes each modality distinct.

Key Insight: Rejecting a Shared Latent Space Assumption

The core insight here is rejecting the assumption that all modalities must reside in the same latent space. Earlier MVAEs operated under the assumption that a single latent representation (often a standard Gaussian) should be used for all modalities. This often led to a latent space that was either:

- Too entangled—forcing incompatible modalities to mix unnaturally, or
- Too loose—failing to capture important relationships between different modalities.

To overcome this, the authors propose learning the latent distribution from the data itself, thereby creating a latent space that respects the nuances of each modality while still allowing information transfer between different modalities.

A Data-Driven Latent Distribution Instead of a Fixed One

Instead of using a fixed target distribution (e.g., Gaussian, as in most cases), MMVM VAE builds a latent distribution in the style of a Mixture of Experts, conditioned on all available modalities during training. Each modality contributes to this learned distribution, making it a soft constraint rather than a hard one, unlike traditional VAEs.

This is a fundamental shift from simple averaging or multiplication of distributions—here, the prior is dynamic, data-dependent, and continuously refined as the model learns the modalities. Essentially, it acts like an adaptive scaffold, shaping the latent space in a way that supports a shared structure without forcing it.

Mathematical Details: Jensen-Shannon Regularization

To the reconstruction loss (how well the model reconstructs the data), the authors add a regularization term that consists of Jensen-Shannon divergences between each modality's approximate posterior $q_\phi(z|X)$, and the learned latent distribution h(z|X), which is computed as an average over all modalities.

What does this mean in practice?

- Each modality's posterior is encouraged to stay close to the learned latent distribution, which is computed from all modalities together.
- However, each modality retains its own structure, preventing the latent space from collapsing into a single, oversimplified subspace (a key argument in the paper).
- The learned latent distribution acts as a dynamic, trainable regularizer, ensuring that representations remain meaningful and useful.

Why MMVM VAE is So Powerful

Ultimately, MMVM VAE does not impose a rigid latent structure (as traditional MVAE models do). Instead, it lets the structure emerge naturally from the data. And this is precisely what makes it so powerful.

Improving Missing Data Imputation

The proposed model also demonstrates strong performance in missing data imputation tasks. MVAEs typically struggle when reconstructing a missing modality from partial input. Why? Because their shared representations are often too rigid—either:

- Over-relying on shared information, resulting in blurry, generic generations, or
- Maintaining completely separate embeddings, preventing meaningful cross-modal interactions.

By contrast, MMVM VAE ensures that each modality's latent space remains informative, even when some modalities are missing. The result? More coherent, contextually relevant reconstructions and sharper fidelity to the expected structure of missing data.

An Interesting Claim: Connection to Contrastive Learning

One particularly intriguing claim in the paper—which I admit I didn't fully grasp—is the connection between MMVM VAE and contrastive learning. The use of Jensen-Shannon divergence as a regularization term aligns the latent distributions across modalities without forcing them into a single subspace.

This is similar to how contrastive learning operates in vision-language models: it ensures that similar inputs produce nearby embeddings while still preserving their differences. However, unlike contrastive learning, which typically requires explicit positive-negative pair construction, MMVM VAE embeds this alignment directly into the generative model itself.

This means that representation learning happens naturally during training, without the need for contrastive sampling tricks or additional objectives.

Trade-offs & Limitations

Of course, trade-offs exist. The model's reliance on data-dependent priors makes unconditional generation—the ability to sample entirely new multimodal data from scratch—non-trivial. Want to generate completely new multimodal data? Too bad—this method wasn't designed for that. But this is not a bug; it's a feature. MMVM VAE isn't trying to be a pure generative model like GANs or diffusion models. Its purpose is structured, interpretable representation learning, where cross-modal dependencies are preserved without imposing artificial constraints.

https://arxiv.org/abs/2403.05300

Mike's Daily Paper – 28.02.25
The FFT Strikes Back: An Efficient Alternative to Self-Attention

Introduction:
Anyone who's been following me long enough knows that I have a soft spot for papers featuring the Fourier Transform or any other periodic transform (such as the Discrete Cosine Transform, DCT). This affinity stems from the five years I spent as a researcher, algorithm engineer, and lecturer in the field of signal processing, specifically in wireless communication systems.

This paper proposes a mechanism that replaces self-attention with a layer that transforms token representations into the frequency domain (i.e., applies a Fourier Transform). The key claim in the paper is that this method has a level of expressiveness (i.e., the ability to model the same types of functions) comparable to Transformers. However, the claims are only partially theoretically proven (a full proof is not provided in the paper).

Why Consider a Frequency-Domain Approach Instead of Self-Attention?

The main advantage of this Fourier-based non-linear transformation approach is, of course, its lower computational complexity. Instead of the O(N^2) complexity of self-attention, the proposed mechanism operates at O(NlogN) - a significant reduction. Here, NNN represents the sequence length. It is well known that the Fast Fourier Transform (FFT) can be computed in O(NlogN) time, and despite the introduction of nonlinear transformations in the frequency domain, the overall complexity of the proposed mechanism remains O(NlogN).

How Does This Work?

1. Fourier Transform on Token Representations
    - The first step is to apply the FFT across the token dimension, meaning each dimension of the token embeddings is transformed separately.
    - Example: If we have 10K tokens, each represented by a 1024-dimensional vector, this results in 1024 independent Fourier Transforms, each of length 10K.
2. Nonlinear Processing in the Frequency Domain
    - Compute the mean of all transformed representations in the frequency space.
    - Pass the result through an MLP (fully connected layers), which outputs a transformed version of the sequence, preserving the original shape (e.g., 10K×102410K in this example).

    - Add the result to a base weight matrix W_base, which consists entirely of ones.
3. Adaptive Reweighting of the Fourier Transformed Representations
    - Perform element-wise multiplication between the transformed representation and the original FFT output.
    - This results in a nonlinear reweighting of the Fourier Transform of the token embeddings, where the weights are learned dynamically.
4. Final Nonlinearity & Inverse Fourier Transform (IFFT)
    - Apply a modReLU (ReLu for complex numbers) to the complex-valued representation.
    - Convert back to the original token space using the Inverse FFT (IFFT).

And the Results? Not Bad at All:

The experimental results show that this FFT-based approach performs competitively while maintaining significantly lower complexity than standard self-attention mechanisms. Definitely an interesting direction for more efficient sequence modeling!

https://arxiv.org/abs/2502.18394

Mike's Daily Paper – 01.03.25
LoRA vs. Full Fine-Tuning: An Illusion of Equivalence

Today marks my 200th daily review since I started writing these summaries nine months ago. To celebrate this milestone, I'll keep things short with a relatively light paper. The paper compares the effects of fine-tuning with LoRA versus full fine-tuning, where all model weights are updated.

Quick Recap: How LoRA Works

In LoRA, we train low-rank matrices that are added to the weight matrices in each layer. A low-rank matrix of size $m \times n$ can be factorized into the product of two smaller matrices: A of size $r \times n$ and B of size $m \times r$ where $r \ll \min(m, n)$ (hence low-rank). Instead of fine-tuning the full model, LoRA updates only the matrices A and B while keeping the original weights W_0 frozen. The modification is applied to specific layers—typically the query (Q) and key (K) projections in the attention layers.

What Did the Authors Compare?

The paper examines how the weight matrices change after fine-tuning—comparing full fine-tuning vs. LoRA fine-tuning. Specifically, they analyze the singular vectors of the trained weight matrices using Singular Value Decomposition (SVD).

Quick SVD Recap:

SVD decomposes any matrixA into three matrices: A=USV^T where: U and V are orthonormal matrices (columns are mutually orthogonal and unit-norm). S is a diagonal matrix containing the singular values. By studying the singular vectors, the authors aimed to understand how much the structure of the weight matrices changes after fine-tuning with LoRA versus full fine-tuning.

Intruder Dimension (InDim): What Gets "Forgotten" in Fine-Tuning?

The authors introduce the concept of Intruder Dimension (InDim) - a dimension in the original weight matrix W_0 that effectively “disappears” after fine-tuning. But how is InDim Defined? For each singular vector of the original weight matrix W_0, the authors try to find a matching singular vector in the fine-tuned weight matrix (after training). The similarity is measured using cosine similarity. If no sufficiently similar singular vector is found, that dimension is classified as an Intruder Dimension (InDim)—meaning it has been lost or forgotten during fine-tuning.

Key Findings

Full Fine-Tuning Removes More Dimensions Than LoRA: The number of InDims is higher in full fine-tuning compared to LoRA fine-tuning. This suggests that full fine-tuning modifies the model’s internal structure more aggressively.

LoRA Rank (r) Affects How Much the Model Forgets. For very low-rank LoRA (r≈1), the number of InDims is relatively low. As r increases, the number of InDims rises, meaning the model is forgetting more of its original structure. However, at higher r values (e.g. r=64), the number of InDims starts decreasing again. This suggests a nonlinear relationship between LoRA rank and model forgetting.

LoRA Leads to More Forgetting Than Full Fine-Tuning: The authors argue that models fine-tuned with LoRA tend to "forget" more of their pre-trained knowledge than models that undergo full fine-tuning. This aligns with previous findings that fully fine-tuned models perform better on out-of-distribution (OOD) data, meaning they generalize better beyond the fine-tuning dataset.

Important Note:

The study focused on fine-tuning encoder-based models, specifically RoBERTa. The conclusions might differ for decoder-based architectures like GPT-style models.

Final Thoughts

This paper challenges the common assumption that LoRA fine-tuning is equivalent to full fine-tuning. The results suggest that while LoRA is computationally efficient, it alters the model’s internal representations differently, potentially leading to more forgetting of pre-trained knowledge. The implications for OOD generalization and optimal LoRA rank selection make this an interesting area for further research.

https://arxiv.org/abs/2410.21228

Mike’s Daily Paper - 02.03.25
An Empirical Model of Large-Batch Training

This is a six-year-old paper from OpenAI researchers, but I found it interesting enough for a quick review. The paper investigates what an optimal large batch size is for training Mini-Batch Gradient Descent (MBGD). What does "optimal" mean here? A batch size that minimizes the number of training examples MBGD needs to reach a target loss value. Naturally, the same examples can be used multiple times throughout training.

For those who need a refresher, MBGD is part of the gradient descent family of optimization methods. It involves splitting the dataset into mini-batches, with each batch consisting of several examples. The model updates its weights once per batch, using the gradient calculated as the average of the gradients of all examples in the batch. In essence, this average serves as an estimate of the overall gradient of the dataset. Each update shifts the model weights in the opposite direction of the gradient, with the learning rate controlling the size of this shift.

The paper proposes a method to determine the optimal batch size (as defined earlier) when using the best possible learning rate—one that minimizes the loss per iteration. It's quite intuitive that the optimal batch size should depend on the model’s parameters, for example, the shape of the loss surface and the values of the gradients.

According to the paper, the optimal batch size can be computed based on the covariance matrix of the gradient, the Hessian of the loss function, and the average gradient vector. This result is derived using a second-order Taylor expansion (in the direction of the gradient), followed by determining the optimal learning rate and then substituting it into the formula to calculate the batch size that yields the maximum possible reduction in loss. The authors then compare this theoretical maximum loss reduction with the actual loss reduction achieved using a given batch size.

The paper highlights several key points. First, the optimal batch size does not depend on the dataset size. Second, the batch size should change dynamically during training because the Hessian, the average gradient, and the gradient covariance matrix typically do not remain constant. Lastly, the paper describes an interesting special case (which doesn't really occur in practice) where the Hessian is simply an identity matrix. In this case, the optimal batch size equals the sum of the variances of all gradient components.

The paper is written in a very clear and accessible way, making it relatively easy to read despite the technical details. Definitely worth checking out!

Paper link

Mike's Daily Paper - 03.03.25
The Geometry of Concepts: Sparse Autoencoder Feature Structure

This Paper explores how sparse autoencoders (SAEs) represent and structure concepts in LLMs. The researchers analyze this structure at three hierarchical scales: atomic, cortical, and galactic. The study makes several comparisons between the embedding space of language models and the structure of the brain, though it is clear that LLMs do not think exactly like humans.

Methodology:

To refresh, SAEs are a tool for studying the interpretability of LLMs. They are trained to reconstruct activations of a specific layer in a model using only a small subset of its features. This sparsity constraint forces the SAE to represent each neuron as a linear combination of a small number of semantic features, with each feature encoding a specific concept (interpretable). In other words, the SAE learns a dictionary of feature vectors (embeddings) in which each neuron is selectively activated for specific semantic patterns.

The researchers use SAEs to extract semantic features from the representations of concepts in LLMs. The study focuses on analyzing the geometric structure of these representations at three scales.

To uncover this structure, the researchers use LDA (Linear Discriminant Analysis) to remove global "distraction" directions in the embedding space, such as word length, which could obscure semantic concept relations. This step is especially important at the atomic level, where analogical relationships become clearer after removing distracting influences.

Atomic Level: "Crystals" and Geometric Patterns

At the smallest scale, the study identifies "crystals"—geometric structures such as trapezoidal parallelograms—within the multi-dimensional feature space. These structures generalize well-known relational concepts such as (man - woman) :: (king - queen). The researchers note that the quality of these geometric patterns improves significantly when global distraction directions, like word length, are removed using LDA.

Cortical Level: Spatial Modularity and "Lobes"

At a medium scale, the study reveals spatial modularity within the feature space of the SAE. Features related to specific domains, such as mathematics and coding, group together to form separate "lobes," similar to functional regions seen in human brain fMRI scans. The researchers

use various measures to quantify the spatial locality of these lobes and find that features appear together with higher density than expected in random feature geometry. These findings suggest that the SAE organizes conceptual features in a way that reflects functional specialization.

Galactic Level: Large-Scale Structures and Eigenvalue Distribution

At the largest scale, the study discovers that the distribution of point cloud data is anisotropic (varying in different directions) and characterized by a Power Law of eigenvalues, with steepest changes (between layers) observed in the middle layers of the network. This indicates that the complexity and variation of data representations are not uniform across layers, with middle layers capturing finer variations in the data. The authors also analyze how the entropy of clusters (in point clouds) changes across different layers of the model, providing insights into the hierarchical structure of concept representations within the model.

https://arxiv.org/abs/2410.19750

Mike's Daily Paper - 05.03.25
Mixtures of In-Context Learners

Modern language models exhibit the ability to perform tasks they were not explicitly trained for by leveraging a few in-context examples—without requiring fine-tuning. This capability is known as **In-Context Learning (ICL)**. I've also seen people refer to it as **Few-Shot Learning**, though this is somewhat misleading, as **Few-Shot Learning** is typically defined as fine-tuning a model on a small set of labeled examples.

How Does In-Context Learning Work?

We provide a language model with **a few examples of the task** in the form of a **prompt**, typically as pairs (x_i, y_i), where:

- x_i is a **question or query**,
- y_i is the **expected response**.

After presenting these demonstration pairs, we input a **new query** x, and the model is expected to generate an appropriate answer **based on the patterns it inferred from the provided examples**.

The Core Research Question

For any given query x, some examples x_i in the prompt will be **more relevant** to it, while others will be **less relevant**.
The key question posed in this paper is: **How can we ensure that the model weighs relevant examples more heavily while downweighting less relevant ones?**

To address this, the authors propose a **weighting mechanism** that assigns **importance scores** to each example in the prompt **relative to the query x.**.

Proposed Method: Learning Example Weights

Given a dataset of labeled examples (i.e., question-answer pairs), the authors **train a model to output a weight w_i for each example in the prompt** relative to the query x. These weights are then used to compute the token distribution for the predicted answer y by:

- Representing the probability distribution of y as a **weighted sum of the log-probabilities of y** conditioned on each individual example pair (x_i, y_i)
- The weights w_i determine the contribution of each example to the final distribution over possible tokens in y.

Training the Weights: Two Approaches

The paper proposes **two different training strategies** for learning these importance weights:

1. **Direct Optimization** – Optimizing a **loss function** based directly on the token representations of x_i and y_i.
2. **Neural Network-Based Weight Prediction** – Training a **separate network** to compute the weights wiw_iwi, and optimizing it by **learning a function** that maps example-query pairs to weight values.

Efficient Inference with Top-K Selection

To avoid the **computational cost** of calculating log-probabilities for **all examples** in the prompt (which can be expensive), the paper proposes an **efficient top-k weighting strategy**. Instead of computing the full weighted sum, only the **top-k most relevant examples** are selected for weighting.

This method is inspired by **Implicit MLE (Maximum Likelihood Estimation)**, which **optimizes a latent model where the latent variable is sampled from a discrete distribution**. The technique is **non-trivial to understand**, and those interested in exploring it further should refer to the references provided in the paper.

📄 **Paper Link:** arXiv:2411.02830 🚀

Mike's Daily Paper - 06.03.25
LYNX: ENABLING EFFICIENT MOE INFERENCE THROUGH DYNAMIC BATCH-AWARE EXPERT SELECTION

I noticed that it's been a while since I reviewed an Paper on MoE - Mixture Of Experts in language models. Recall that MoE is a method designed to optimize inference in terms of computational load (i.e., fewer calculations). The model is trained to activate only part of the model (specific experts) for each token, where each expert is (usually) a sub-network of the FFN (in fact, it is usually as sub-matrix of the weight matrices in the FFN) within the transformer mechanism. In practice, this makes it possible to reduce the amount of computation per token, which may allow the activation of LLMs of enormous size (only part of the model each time). In

addition (according to several studies), this method makes it possible to learn "more complex functions" because each token may be calculated differently (with a different subset of experts).

The experts are selected by a routing network, where it is trained to compute a non-negative score for each expert. Scores are actually "probabilities" of selecting each expert (there is softmax at the end). Usually, k experts with the highest scores are selected in each layer for each token out of N experts, where k < N. The model is trained to balance the utilization of each expert, with the goal that each expert will be utilized equally in the training dataset (aggregative level). Usually, there is a regularization term on the weights of the routing network, for example in the form of negative entropy or the sum of squares.

The paper proposes a method for optimizing memory consumption for inference of transformer models with MoE when they are activated in batches of queries (several inputs). The proposed approach is based on several empirical observations made by the authors:

- The distribution of the frequency of expert activation within the batch is not uniform, meaning there are experts that are activated more and there are those that are activated less.
- The computational density (arithmetic intensity), which is the ratio between the amount of flops and the amount of memory accesses, decreases when the number of experts increases in the decode phase (i.e., prediction). This makes this phase memory-bound, which increases the latencies.
- The tokens are not very sensitive to their experts beyond a few experts (from top-k) with the highest scores. That is, it is possible to "activate only the experts" without significant damage to performance.
- Not all tokens are equal, meaning there are tokens that are more sensitive to the use of some of their experts and there are those that are less. The authors claim that it is possible to infer the level of sensitivity of the token from the routing network scores for it.
- The prefill phase (prompt processing) is more sensitive to the replacement of experts than the decode phase (generation).
- The sensitivity to the replacement of experts varies between the layers of the model, where the middle layers are the most sensitive to it.

The authors propose to take advantage of these observations in the following way (there are several variations, I will describe the main method):

- All experts are used in the prefill phase (which is compute-bound).
- Sensitive and less sensitive tokens (low and high confidence) are identified in the batch. Then the experts of the less sensitive tokens are filtered.
- The experts that are most used for the batch are selected and the rest are filtered.
- Only the remaining experts are activated for all tokens (top-k). A second option (less damaging to performance) - is to activate all experts for sensitive tokens and only those that remain for less sensitive tokens.

This method makes it possible to increase computational density for the decode phase and make it less memory-bound without significant damage to performance.

https://arxiv.org/abs/2411.08982

Mike's Daily Paper - 07.03.25
Number Cookbook: Number Understanding of Language Models and How to Improve It

Introduction

I've always argued for using simple tools like calculators or code for arithmetic calculations, but people insist on using LLMs for that and there's a price to that.

The paper provides an in-depth analysis of the numerical understanding and processing ability (NUPA) of LLMs. The authors introduce a structured benchmark to rigorously evaluate LLMs across 4 numerical representations and 17 task categories, leading to 41 distinct test cases. By doing so, the work highlights fundamental deficiencies in modern LLMs when handling numerical reasoning tasks. This review dissects the paper's contributions, evaluates its methodology, and discusses its implications for improving LLM numerical reasoning.

The paper's central premise is that numerical competence is not an emergent property of large-scale pretraining but rather a distinct capability requiring targeted interventions. The failure of LLMs in seemingly trivial numerical tasks like ordering floating-point numbers or performing modular arithmetic contradicts their prowess in complex symbolic reasoning. The authors argue that despite advancements in large-scale training, fundamental numerical processing remains a bottleneck, potentially affecting the integrity of more advanced mathematical and analytical reasoning tasks.

A Benchmark for NUPA

Yang et al. present a carefully designed benchmark that categorizes numerical tasks based on representations such as integers, floating points, fractions, and scientific notation. The granularity of these tests is a notable improvement over existing mathematical reasoning benchmarks, which often conflate problem-solving heuristics with core numerical understanding.

By formulating the benchmark around fundamental arithmetic, digit comprehension, and conversion tasks, the authors ensure that their evaluation isolates numerical understanding from broader reasoning capabilities. This structured approach allows for precise measurement of weaknesses in LLMs and provides a roadmap for future improvements. The benchmark's basis in primary and secondary school curricula ensures that the included tasks are representative of real-world numerical understanding.

Empirical Evaluation of Leading LLMs

The study rigorously evaluates models such as GPT-4o, LLaMA-3, and Qwen2, revealing that even state-of-the-art models falter on basic numerical tasks, especially as complexity or input length increases. The observed performance degradation across tasks like modulo operations and digit alignment underscores a crucial blind spot in current LLM architectures.

This evaluation is particularly striking in its revelation that numerical errors persist even in tasks where models achieve high scores on broader mathematical benchmarks. The authors systematically analyze how different numerical representations impact performance, exposing a drastic decline when moving from integer-based tasks to floating-point or fraction-based ones. This insight is invaluable, as it suggests that current training paradigms fail to generalize numerical competence beyond integer-based arithmetic.

Techniques for NUPA performance improvement

The authors explore 3 core methodologies for improving NUPA: modifying tokenization strategies, fine-tuning models on numerical tasks, and leveraging positional encodings (PEs) and digit alignment techniques. Surprisingly, while naive fine-tuning significantly improves performance, specialized techniques such as alternative tokenization and index hints often disrupt model behavior rather than enhance it.

The tokenization experiments provide particularly novel insights. One-digit tokenizers outperform multi-digit tokenizers, contradicting the prevailing notion that longer tokens improve efficiency. The findings suggest that LLMs struggle with numerical alignment when tokens encompass multiple digits, likely due to the way transformers process sequences. Furthermore, PE modifications designed to enhance numerical learning often yield negative results, indicating that the interaction between tokenization and positional information in numerical tasks is non-trivial.

Fine-tuning experiments indicate that substantial improvements in NUPA can be achieved through targeted post-training, but these benefits do not necessarily translate across all numerical tasks. The inability of fine-tuned models to significantly improve digit retrieval tasks, for instance, suggests that fundamental numerical encoding mechanisms require rethinking at the architectural level rather than just via dataset augmentation.

Chain-of-Thought (CoT) Analysis for Numerical Tasks

The authors experiment with rule-following CoT (RF-CoT) to evaluate whether explicit step-by-step reasoning mitigates numerical errors. Although CoT methods boost accuracy, their drawbacks—higher inference time and context window limitations—make them impractical for everyday numerical reasoning.

The experiments reveal that while CoT-based reasoning improves accuracy in structured calculations like multi-digit multiplication, it quickly becomes computationally prohibitive. The cost of generating intermediate reasoning steps outweighs the accuracy benefits, making CoT infeasible for real-world applications requiring large-scale numerical inference. Additionally, the

study identifies a performance ceiling beyond which additional reasoning steps do not improve accuracy, reinforcing the idea that fundamental representation and processing improvements are necessary rather than procedural workarounds.

Conclusion

Yang et al. make a significant contribution by analyzing and benchmarking NUPA in LLMs. The work exposes fundamental limitations and provides empirical evidence for effective and ineffective strategies to improve numerical processing. While the study's findings highlight persistent challenges, they also offer a valuable roadmap for the AI research community to enhance numerical reasoning in future LLM iterations.

Overall, the paper underscores the need for dedicated numerical processing mechanisms within LLMs. As models become more capable at complex reasoning tasks, their inability to handle simple numerical operations becomes an increasingly critical issue. This research lays the groundwork for future advancements in numerical representation learning, efficient tokenization strategies, and hybrid approaches that combine statistical learning with explicit numerical reasoning paradigms.

Or just do these calculations on the calculator or with python code….

https://arxiv.org/abs/2411.03766

Mike's Daily Paper - 09.03.25
THE SUPER WEIGHT IN LARGE LANGUAGE MODELS

It's quite unbelievable, but LLMs with billions or even dozens of billions of parameters can fail to deliver high performance if you remove even a single weight. This surprising finding applies to at least some of these powerful models.

This paper investigates a highly specific and unexpected characteristic of LLMs: the existence of "super weights(SWs)." The authors move beyond the now-established observation that LLMs harbor influential outlier parameters, presenting evidence that a solitary scalar weight can be disproportionately critical to model function.

The central finding is that pruning a single SW can precipitate a catastrophic decline in LLM performance. This drastic effect is manifested as a sharp increase in perplexity and a reduction of zero-shot accuracy to near-random levels. What's particularly noteworthy is the stark contrast between the impact of SW removal and the comparatively minor effect of pruning other, even larger magnitude, outliers.

The paper gives an interesting example of the influence of removing a SW for the prompt: “Summer is hot. Winter is ”. The correct next token should be “cold”, with a strong semantic meaning. With the original model with SW, it correctly predicts the next token “cold” with a high

probability 81.4%. However, when the SW is removed, the model's top prediction is a stopword "the" with a non-confident low probability of 9.0%. This indicates that the SW is essential for the model to make a correct and confident prediction of meaningful words.

The paper doesn't just document this phenomenon; it also explores the underlying mechanisms. The authors link SWs to the generation of "super activations," which are large activation outliers that propagate through the model. They further provide a data-free methodology for identifying these SWs, leveraging the detection of spikes in activation distributions.

Moreover, the study investigates the implications of SWs for quantization. The presence of outliers, including SWs and their induced super activations, poses a significant challenge to effective quantization, as these outliers can skew the quantization process and lead to substantial information loss. The authors demonstrate that preserving super outliers (both weights and activations) can enhance the efficacy of round-to-nearest quantization, even allowing for the use of larger block sizes in weight quantization.

This is achieved by holding out super outliers during quantization and restoring their values afterward, mitigating the adverse effects of these extreme values on the quantization of other parameters. By addressing the challenges posed by super outliers, the proposed approach enables the application of simpler and more efficient quantization methods, facilitating model deployment in resource-constrained settings.

This work makes a strong case that SW are not merely isolated anomalies but rather integral components that play a vital role in shaping LLM behavior and efficiency, with significant ramifications for model compression and deployment. The paper's contribution lies not only in the identification of SWs but also in characterizing their functional role within LLMs. The authors dissect how these weights exert their influence, connecting them to the emergence and propagation of super activations.

https://arxiv.org/abs/2411.07191

Mike's Daily Paper Review – 11.03.25
Beyond Matryoshka: Revisiting Sparse Coding for Adaptive Representation

A short review of a paper that generalizes a method for generating low-dimensional representations of data called Matryoshka embeddings. What makes this method special? It allows training representations at multiple dimensions simultaneously. That means during training, embeddings of different sizes (e.g., 8, 16, 32, 64, and 128) are learned at the same time.

The method assumes a labeled dataset of pairs (x, y) where x is a data sample and y is its label. Matryoshka embeddings train a deep network with a final classification head that maps the data representation to its corresponding label. The unique aspect of Matryoshka is that it trains multiple projection vectors (along with the model itself) into the label space, where each projection vector takes the first m_i components of the embedding vector (the model's last layer). In the example I mentioned earlier, it simultaneously trains mapping vectors of sizes 8,

16, 32, and 64. The loss function is the sum of the losses for all these vectors—so in addition to training the model itself, we train four vectors of sizes 8, 16, 32, and 64.

The paper being reviewed repurposes this interesting approach for improving sparse data representation by modifying it in two ways (essentially tweaking the loss function).

Sparse Auto-Encoder (SE): Instead of the original method, which directly maps the model's representation to labels, an SE is trained to project the data representation into a high-dimensional but sparse space and then reconstruct it back into the original representation space.

Important note: The model itself is not trained here - only the mapping vectors (of the SE).

Contrastive Loss: This is added to push representations of different categories further apart while pulling together representations of data samples from the same category.

So, what's the goal of the SE in this case? Unlike the original Matryoshka method, which trains only the first elements of the representation vector, here we extract the top-k components of the vector after encoding. The decoder is then trained to reconstruct the original vector using only the top-k components of the encoded representation.

The Known Issue with SE and How They Address It:

A well-known problem with SE is that certain components of the encoded vector become "dead", meaning they take on very similar values across all data points, making them uninformative. To address this, the authors propose two solutions:

- Adding multiple k-values to the top-k encoder in the loss function (instead of a single k-value). This trains embeddings at multiple sizes, similar to Matryoshka, though the main goal here is to create sparse embeddings, not just multi-size embeddings.

- Adding a term to balance reconstruction error between the top-k largest components and the "dead" components (the smallest values in the encoded vector). I couldn't fully grasp why this helps, but that's their approach.

Additionally, as mentioned, they also introduce the contrastive loss described earlier.

Final Thoughts:

While "Matryoshka" appears in the paper's title, its connection to the original Matryoshka embeddings is quite loose. That said, the paper itself is still quite interesting.

Read the full paper: https://arxiv.org/pdf/2503.01776

Mike's Daily Paper – 11.03.25
Transformers are Universal In-Context Learners

Today, we will briefly review a heavy theoretical paper investigating the expressivity of deep transformers. Transformers are deep architectures that define "in-context mappings," which enable predicting new tokens based on a given set of tokens. Note that in-context here has a slightly different meaning than in in-context learning, which refers to transformers' ability to learn tasks they were not trained for based on a few prompt examples (at least to the best of my understanding).

The authors prove that deep transformers (with many transformer blocks) are universal approximators, meaning they can approximate any continuous in-context mapping over token distributions to arbitrary precision. Furthermore, these results hold for both bidirectional attention mechanisms (as in encoders) and causal attention mechanisms (as in decoders), while maintaining a fixed embedding dimension independent of the number of tokens.

The Measure-Theoretic Approach (Finally Found a Use for It in DL Papers!):

The proposed approach is based on measure theory, where token sequences are represented as probability distributions in the embedding space. This enables the use of tools from functional analysis (flashbacks from nearly 30 years ago in my undergraduate studies) and optimal transport theory (which I wrote about extensively back in the Wasserstein GAN days) to establish the universal approximation property of transformers. A key technical contribution is the reinterpretation of the attention mechanism as an operator on distributions, allowing the application of Stone–Weierstrass theorem (a hardcore generalization of the Weierstrass approximation theorem from Calc II, I believe). This fundamental result in approximation theory states that any "nice" function can be approximated by a sufficiently rich family of simpler functions (though the theorem is quite intense, involving functions on Hausdorff spaces and such).

Measure-Based Representation of In-Context Learning:

A central novelty of the paper is representing the attention mechanism as an operator on probability distributions, rather than on finite token sequences. This allows for a uniform analysis of in-context learning, independent of the number of tokens in the sequence. Instead of working with finite sets of token embeddings, the authors define a space of probability distributions over a compact subset of Euclidean space (the embedding space). A distribution assigns weights to different token embeddings, effectively transitioning from learning over a finite set of tokens to a continuous and infinite representation.

Formally, a token sequence can be represented as a discrete probability distribution, consisting of a weighted sum of Dirac delta functions, each centered at an individual token embedding. As

the number of tokens increases, these distributions converge to continuous distributions. This formulation allows proving results that hold for any number of tokens, including infinitely many.

Defining Attention as an Operator on Measure Spaces

A typical transformer layer consists of two components:

Multi-head attention mechanism, which updates token representations by computing their interactions.

Feed-forward networks (FFN), which process each token independently after the attention step.

The authors reformulate the attention mechanism as a mapping operating on distributions of tokens. Instead of summing over a finite set of tokens, attention is defined as an integral operator over a space of distributions, effectively making tokens a continuous structure. This formulation is particularly important because it enables defining continuity and smoothness of in-context mappings using Wasserstein distance (a special case of which is the Earth Mover's Distance), which measures the distance between probability distributions. A function is continuous in the Wasserstein sense if small changes in the input distribution lead to small changes in the output distribution. This property ensures that the mappings created by transformers are stable to variations in the learning context.

Proving Universality: Approximating In-Context Mappings

The key results of the paper prove that transformers are universal approximators for in-context mappings. The authors show that for any continuous function mapping token distributions to outputs, there exists a deep transformer that can approximate it to arbitrary precision. A crucial part of the proof involves constructing fundamental functions in context, which serve as building blocks for approximating any general function in the previously defined spaces. These fundamental functions are simplified versions of transformer layers that capture the essential principles of attention mechanisms.

A fundamental function consists of three components:

A linear transformation of the token embedding (affine mapping).

A nonlinear interaction that considers the distribution of all tokens.

A context-dependent adaptation, enabling the model to "learn in context."

These functions operate similarly to single-head attention mechanisms but are mathematically more tractable. The authors prove that by stacking multiple layers of these functions, one can construct deep transformers capable of approximating any function in context.

Using the Stone–Weierstrass Theorem

To prove universality, the authors show that their set of fundamental functions satisfies the conditions of the Stone–Weierstrass theorem, a key result in functional analysis. They demonstrate that these fundamental functions form a sufficiently rich function class, ensuring that deep transformers can approximate any in-context mapping.

Wrapping up:

This paper provides a mathematical framework for proving the expressivity of transformers in learning in-context mappings, leveraging functional analysis, measure theory, and optimal transport theory. The results show that deep transformers can approximate any context-dependent function, independent of the number of tokens in the context window.

https://arxiv.org/abs/2408.01367

Mike's Daily Paper - 14.03.25
A Survey on Kolmogorov-Arnold Network

Introduction:

Do you remember KANs? This stands for Kolmogorov-Arnold Networks, which caused quite a stir back in the day, but the buzz has since faded. It turns out that many studies have emerged since then on this fascinating topic. The paper discusses various extensions and modifications to the basic KAN architecture. These include adaptations for time series analysis, graph data processing, and solving differential equations. These modifications typically involve the integration of special components or constraints within the basic KAN architecture to better address the specific requirements of these domains.

KANs represent a paradigm shift in neural network design, transitioning from fixed activation functions to learnable functions known as B-splines. This was inspired by the Kolmogorov-Arnold representation theorem, which states that any continuous function of multiple variables can be represented as a composition of functions of a single variable. By using functions represented by splines (a combination of polynomials over a specified interval), KANs offer improved flexibility and the potential for higher accuracy in function approximation. This leads to enhanced model interpretability, as the learned univariate functions can be more easily analyzed.

KAN Networks for Various Domains:

Now let's describe several extensions of KANs for different domains. For time series analysis, temporal KANs (T-KANs) integrate memory mechanisms, similar to RNNs and LSTMs, to efficiently handle these sequences and the long-term dependencies within them, showing excellent performance in multi-step forecasting tasks. Additionally, changes such as gated

connection mechanisms, similar to LSTM and GRU, allow KANs to dynamically adjust activation functions (represented by the splines) based on task complexity, improving efficiency without the need for extensive regularization.

For graph data, graph-based KANs (GKANs) were developed to enhance semi-supervised node classification by improving information flow between nodes, outperforming traditional Graph Convolutional Networks (GCNs). These KAN-based architectures improve node representation learning and enhance regression model accuracy in graphs arising from social networks and molecular chemistry. GCNs operate by aggregating and recursively transforming feature information from local neighborhoods within a graph, capturing both node features and graph topology.

However, GCNs rely on fixed convolutional filters, which limit their flexibility in handling complex and heterogeneous graphs. To address this limitation, GKAN introduces two main architectures: Architecture 1, which aggregates node features before applying KAN layers, enabling learnable activation functions to capture complex local relationships, and Architecture 2, which places KAN layers between node embeddings in each layer before aggregation, allowing for dynamic adaptation to changes in graph structure. This improvement enables GKANs to adapt dynamically to changes in the graph structure, providing a more adaptive approach to graph-based learning.

For solving differential equations, physics-based KANs (PIKANs) were adapted as an interpretable and efficient alternative to physics-informed neural networks (PINNs) based on MLPs. Here, PIKANs use a

grid-dependent adaptive structure, making them suitable for applications requiring precision, such as flow dynamics and quantum mechanics, where dynamic basis functions help capture complex physical processes with improved accuracy and computational efficiency.

The authors also discuss the challenging optimization of KANs due to the nonlinear nature of the spline parameters in high-dimensional spaces, which are frequently encountered.

Summary:

KANs use B-splines for parameterizing functions of a single variable, making them learnable and enabling smooth transitions between different intervals with improved local fitting of data. The optimization process involves adjusting spline parameters, such as control points and knots, to minimize errors between predicted and true outputs, allowing the model to capture complex data patterns. However, this process is complicated due to the highly nonlinear parameter space structure, the curse of dimensionality, and the increased computational overhead resulting from the flexibility of the learnable splines.

Mike's Daily Paper - 15.03.25
Generative Representational Instruction Tuning

I stumbled upon this paper pretty much by accident—during a random conversation with a typical LLM about contextualized embeddings and how they're constructed. It's a fairly light read, so I figured that if I already spent 5 minutes reading it, I might as well spend another 10 reviewing it.

The paper proposes a method that combines instruction tuning (InTn) for generation with InTn for constructing contextual data representations.

- Generative Instruction Tuning (Generative InTn) is quite straightforward - it trains the model to follow user instructions (e.g., for chatbot applications).
- Representational Instruction Tuning (Representational InTn), on the other hand, trains an encoder model to build vector representations of text in a way that depends on user instructions (which is quite similar to contextualized embeddings).

There are plenty of papers discussing how to train a model to handle each of these tasks separately—but this paper proposes a method to train the same model to perform both (not simultaneously, though).

The method itself is simple: combine two loss functions - one for Generative InTn and one for Representational InTn. Each task is assigned a pre-trained task-specific head (a few transformer blocks, as far as I understand). For the first task (generative instruction tuning), the authors use the standard loss function for generative language models—predicting the next token, but only for the response. For the second task (representational instruction tuning), they use a contrastive loss (pretty standard for these tasks), which tries to pull together embeddings of a question and its correct answer while pushing apart embeddings of the same question paired with an incorrect answer.

The text representation is computed bidirectionally (encoder-based), where the final embedding is the mean of all token embeddings in the text. Naturally, each task gets its own prompt.

That's it—a quick review, just as promised.
https://arxiv.org/abs/2402.09906

Mike's Daily Paper - 17.03.25
JanusFlow: Harmonizing Autoregression and Rectified Flow for Unified Multimodal Understanding and Generation

It's been a while since I reviewed a paper on multimodal generative models. These models are trained not only to generate data of multiple types (in the case of JanusFlow, natural language and images) but also to perform tasks involving the understanding of relationships between these modalities. For example, a multimodal model in the domain of language and images should be able to answer questions about an image.

The model consists of a main model (called LLM) and several encoders and decoders designed to represent data from different modalities and convert these representations into actual data (decoders). All the models in the paper are Transformer-based, which is not surprising.

The paper proposes a method to train such a multimodal model (referred to as LLM in the paper), with an interesting detail—using different encoders for language and images (whereas most multimodal models use the same backbone model). Essentially, during training, the model learns to predict the tokens of an answer given a prompt, where both the prompt and the answer can be either a visual token (a representation of an image patch) or a regular token (a sequence of letters). Additionally, the prompt can be a combination of visual tokens and language tokens in the visual question answering task.

Moreover (not explicitly mentioned in this paper but done in other multimodal models), the model is also trained on textual data only (similar to pretraining a regular language model).

A few details about the different models (besides the LLM) appearing in the paper:
For linguistic data, tokens pass through a trained encoder (called und enc)—after which the tokens go through a trained linear layer. For visual data, there is a standard, untrained encoder based on VAE, followed by another trained encoder. Since the generative model for images is a diffusion model, the use of a VAE (a separate component in generative diffusion models) should not be surprising. Additionally, as mentioned, there are two trained decoders that receive the representations built by the LLM.

The paper proposes a three-stage training method, where in each stage more and more models (including the LLM) are "unfrozen", and in the final stage, all models are trained except for the VAE.

The diffusion models in the paper are Rectified Flow (RF) based, which attempts to map data from a simple distribution (Gaussian) to the data distribution in a straight path—meaning the trajectory between x_0 (Gaussian) and x_1 (data) is linear. In other words, every point x_t along this trajectory is a convex combination of x_0 and x_1. Essentially, the diffusion model is trained to estimate the constant velocity v (which equals x_0 - x_1 for every point x_t along the trajectory). Sampling is performed by solving a differential equation describing the progression of x_0 to x_1 with velocity v (Euler method). The diffusion model trained in the paper is latent-based.

An interesting detail about the paper: one term in the loss function of the diffusion model penalizes the mismatch between the noisy internal representation (computed at the intermediate layers of the model) and the clean image representation computed by a strong encoder (understanding encoder). And of course, classifier guidance is used in training the diffusion model (a classic approach in the generative diffusion models).

A well-written and quite clear paper—highly recommended!
https://arxiv.org/abs/2411.07975

Mike's Daily Paper – 19.03.25
EFFICIENTLY LEARNING AT TEST-TIME: ACTIVE FINE-TUNING OF LLMS

Recently, the most popular method for adapting language models to a specific task is in-context learning (ICL). Essentially, we provide the model, within the prompt, with a few examples of how to perform the task, and the model "learns" how to execute it without any changes to its weights. ICL is possible due to the adaptive nature of transformers (the attention mechanism within them), which manages to "update its computation" as a function of the input.

The paper discusses a different method for adapting a model to a given task at test time (the paper somewhat mixes the concept of test and inference), which involves a light fine-tuning of the model based on the prompt it receives. Unlike ICL, the proposed method—SIFT (Selects Informative data for Fine-Tuning)—does change the model's weights (performs a single gradient descent step). In fact, SIFT (by the way, there is also a method with this name in image processing from the pre-neural network era) proposes a method for selecting examples from the dataset for fine-tuning the model on a given prompt.

The authors argue that selecting examples closest to the prompt in latent space based on cosine distance or inner product (nearest neighbors or NN) is suboptimal and may retrieve redundant examples that harm fine-tuning performance. Instead of retrieving the most similar examples to the prompt, SIFT selects those that provide the most new information, thereby achieving better model adaptation with minimal additional computations.

Uncertainty Estimation for Guiding Fine-Tuning – Why Is This Important?

Many fine-tuning methods rely on retrieving similar examples based on cosine similarity or Euclidean distance. However, this approach is flawed: it does not distinguish between relevant and redundant data. Two very similar examples may contain the same information, and therefore, one of them does not contribute to the fine-tuning result. To solve this, the authors propose a method for estimating the model's uncertainty in its response after fine-tuning.

- If the model is very confident in its answer after fine-tuning, adding an example will not significantly affect the result.
- If uncertainty is high, smart selection of examples can greatly improve model performance.
- The challenge is efficiently finding these examples.

Measuring Similarity in Latent Space Using a Kernel Function

As mentioned, the basis of SIFT's selection method is measuring similarity between examples in latent space. To quantify this similarity, the authors use a kernel function, which is defined as the inner product between the latent representations of the examples.

This function takes two sequences and returns a similarity score—high for similar sequences and low for different ones. Using this kernel function, they construct a kernel matrix for the selected fine-tuning examples and the prompt itself. They then define a surrogate model, whose purpose is to estimate the performance of the LLM after fine-tuning on the selected examples.

Using this model (which is somewhat mathematically heavy), they estimate the model's uncertainty after adding an example xx from the dataset to the set of examples used for fine-tuning. Ultimately, they select the example that minimizes uncertainty for the prompt and add it to the set of examples.

In Simple Terms, the Proposed Approach Balances Two Opposing Considerations:

- Relevance: The selected examples should still be relevant to the prompt.
- Diversity: The examples should not contain overlapping and redundant information.

Instead of selecting examples all at once, SIFT selects each example gradually, using the kernel function to determine its added value. The essence of proposed method can be summarized in the following manner:

- The essence of the proposed method can be summarized as follows:
- If a new candidate is too similar to previously selected examples, it is rejected, as it does not add new information.
- If the candidate is relevant but contains new details, it is selected to reduce uncertainty.
- If the candidate is completely unrelated to the prompt, it is excluded from the process.

https://arxiv.org/abs/2410.08020

Mike's Daily Paper - 20.03.25
Softmax is not enough (for sharp out-of-distribution)

This paper proposes a method to improve generalization performance for transformer models from a rather unexpected angle. The authors propose a method to deal with what is called dispersion of attention weights in transformers. This manifests, for example, in the inability (according to the paper) of transformers to focus attention weights on a small number of tokens (relative to sequence length). This is important, for instance, in tasks like finding maximums in a

given number sequence or "needle in a haystack" questions where the model is asked to find a short unrelated passage in a relatively long text.

The authors claim that one of the main reasons for these problems is the high dispersion(making all of them more or less equal) of attention weights in the transformer mechanism. These weights are calculated with a softmax function that "normalizes" the inner products of K and Q vectors for all sequence tokens. According to the paper, the problem is related to the fact that for long contexts, softmax, especially in deep transformers, has a "tendency to smear the softmax output."

One way to deal with this phenomenon is to lower the temperature, but this may increase the likelihood of errors in cases where the logit (unnormalized attention weight) of the correct token is smaller than the maximum logit. To address this phenomenon, the authors proposed a new version of softmax where the temperature depends on token entropy.

They trained a model for cases where the logit of the correct token is not maximal, with the goal of maximizing the sampling probability of the correct token (after the attention mechanism and FFN). The model's purpose was to compute an optimal temperature value as a function of the entropy of unnormalized attention weights. The temperature formula turned out to be inverse (1 divided by a polynomial of power 4 with learned coefficients). It is worthwhile to note that the temperature is calculated during inference depending on token entropy according to this model.

The authors empirically showed that adaptive temperature reduces dispersion of attention weights. Although the optimal adaptive temperature decreases with increasing logit entropy, it causes fewer model errors compared to when it is hardcoded (namely hyperparameter).

https://arxiv.org/abs/2410.01104

Mike's Daily Paper - 21.03.25
LLMs learn governing principles of dynamical systems, revealing an in-context neural scaling law

This paper caught my attention because it mentions language models and dynamical systems, two topics I've liked since the good old state-space model (Mamba) days. The paper claims that LLMs exhibit strong performance in understanding various types of dynamical systems,

including stochastic, chaotic, and continuous systems, all without any fine-tuning—just some prompt engineering, and suddenly your LLM understands dynamical systems.

Recently, several papers have explored using LLMs to model dynamical systems by generating new samples from them. The logic here is fairly simple: if an LLM can generate data following the distribution induced by a dynamical system, it likely understands the system itself.

However, the authors take a more direct approach—they demonstrate that an LLM can explicitly construct the probability distribution of a dynamical system, assuming it follows a Markov process. This means that the distribution of the next state at time t+1 depends only on the state at time t and not on previous states.

For a discrete system, this distribution is represented by a transition probability matrix, which contains the conditional probabilities of the next state xt+1x_{t+1}xt+1 given the current state xtx_txt, for all possible values. In continuous systems, such a matrix can be constructed by discretizing the state space.

The paper shows that LLMs can construct these transition matrices quite effectively, especially when the number of possible states is relatively small. The difference between the LLM-predicted distribution and the ground truth distribution is measured using Bhattacharyya distance, which I've only encountered once before in deep learning papers. The paper also reports strong results using other divergence measures, such as Jensen-Shannon (JSD) and KL divergence.

The authors propose a tricky method to construct these transition matrices with an LLM—if you want to dive deeper, check out the Hierarchy-PDF algorithm section.

That's it for today, a short one…
[Paper link]

Mike's Daily Paper - 22.03.25
Physics in Next-token Prediction

This paper is far from ordinary. It starts with its title—how could next-token prediction (NTP) possibly be related to physics? It turns out the connection exists, and it runs through information theory. Those who know me are aware of my deep interest in the informational aspects of machine learning, such as how models compress information, how knowledge is stored in trained models, and related topics. This paper directly addresses these ideas, and while it doesn't involve overly complex math, it is quite deep (not sure I fully absorbed it myself! 😅).

Let's start with Shannon's theorem, with a slight twist: it states that the number of bits required to transmit the next word x_{t+1} after transmitting t words is equal to the conditional entropy H of x_{t+1} given the previous words, or equivalently, its self-information I. In practical terms, this entropy is simply the log-probability of the conditional probability of x_{t+1} given the previous words. From this, it follows quite naturally that the total number of bits required to transmit an entire dataset D is the sum of these conditional entropies over all words in D.

Now, let's assume we have a model trained to predict the next token given its context (i.e., previous tokens)—a typical language model. The number of bits required to transmit the same dataset D is still calculated using the same formula—the sum of conditional entropies. However, this time, since the model itself is making the predictions, we likely need fewer bits to encode D. But why does this happen? Where does the difference in bits between encoding D without the model and encoding it with the model go?

Since information cannot be lost, the assumption is that the model has absorbed it—it has learned. The paper refers to this stored information as the model's effective information about the dataset D (or the task). It also introduces η, which represents the capacity of the model, defined as the ratio of the model's effective information to the number of parameters in bits. Additionally, an interesting observation is made: the number of bits required to transmit D using the model is simply the cross-entropy loss of the model on D multiplied by |D| (the number of tokens in D).

Tying these quantities together leads to the first law of thermodynamics for LLMs, as defined in the paper:

$$\eta N = |D|(H - L)\eta N$$

where N is the number of model parameters, L is the cross-entropy loss of the model on D, and H is the initial entropy of D. During training, H and N remain constant, while |D| represents the number of tokens the model has "seen." This suggests that training a model is essentially a process of compressing the dataset D and transferring its information into the trained model.

The paper also defines a second law of thermodynamics for LLMs, which describes the minimum energy required to transfer information from D to the model. This energy is proportional to N and η and even includes a temperature term T (not to be confused with LLM temperature) as well as Boltzmann's constant k. I'll admit—I didn't quite grasp the full meaning of those last two (T and k).

Based on this theoretical framework, the authors derive interesting insights into model training and even compare their formulated laws to scaling laws in language models. If you're into this kind of stuff, I highly recommend diving in—a fascinating read.

Paper link

Mike's Daily Paper Review – 26.03.25
DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining

Today's paper belongs to a field I wasn't familiar with before, so there's a chance I might have made some mistakes in my review—despite my best efforts. The paper discusses optimization strategies for training language models when dealing with datasets from different domains.

Mathematically speaking, the authors propose a weighting strategy for different datasets during training. Given d datasets, the goal is to find a d-dimensional vector α of non-negative numbers

summing to 1, where α_k represents the probability of sampling an example from dataset D _k . This means the training set is constructed in two steps: select a dataset based on probabilities from α and sample an example from the chosen dataset.

A simple approach would be to set α_i proportional to the dataset size meaning larger datasets are sampled more frequently. Another option is uniform sampling, where each dataset is chosen with probability 1/d. Some methods select α based on dataset quality, favoring high-quality datasets over lower-quality ones. But how do we choose α to maximize model performance? This is the key question the paper attempts to answer.

Brute-Force Search vs. Smarter Approaches
One naive solution is brute-force search: try different values of α, train a model for each, and compare results. But for large models and a high number of datasets d, the computational cost becomes prohibitively expensive. So, can we do something more efficient? The answer is yes, and that's exactly what the authors propose.

Proposed Approach: Distributionally Robust Language Modeling (DRO-LM)

First, they train a small model M_ref with some initial dataset weights α_0 (e.g., uniform sampling). The method is inspired by distributionally robust optimization (DRO) and aims to minimize the worst-case error across all valid dataset weight vectors α. The error is defined as the difference between the token log-likelihood of the trained model and M_ref , averaged over tokens in each dataset. If you noticed a minimax problem, you're absolutely right. The optimization process alternates between:

- Gradient ascent over α (to maximize the worst-case error).
- Gradient descent over model parameters (to minimize the model's loss given the chosen α).

Iterative Optimization:

Over multiple iterations, batches of training examples are randomly sampled while solving the minimax problem. The final dataset weight vector α is obtained by averaging the optimized α values from all iterations.

Final Training Step:

A large-scale model is then trained using the learned dataset mixture α, leading to better generalization and training efficiency.

Final Thoughts
This method provides a computationally efficient way to optimize dataset weighting, significantly improving pretraining speed and performance. Instead of brute-force searching for the best dataset mixture, it learns an optimal weighting through adversarial minimax optimization.

Hope I managed to explain it clearly!

📄 Read the paper here: https://arxiv.org/abs/2305.10429

Mike's Daily Paper Review – 28.03.25
UniMax: Fairer and More Effective Language Sampling for Large-Scale Multilingual Pretraining

I continue my exploration of papers on optimizing data selection for model training, particularly for language models, which we all appreciate. The problem can be formulated quite simply: given a set of datasets D = {D_1, ..., D_n} the goal is to choose an "optimal" combination of these datasets for training a model under a budget B, which represents the total number of tokens/symbols (referred to as symbols in the paper) that the model "sees during training." This can also be translated into FLOPs, given the model's architecture. In simple terms, we want to understand how to sample data from D to achieve the best-performing model after training within the given budget B. Of course, "best" can be defined in multiple ways, but we will focus less on that and more on how to balance examples from different sources (datasets) when training the model.

The paper discusses the scenario of training multilingual models when data is available in multiple languages. A straightforward approach is to allocate an equal budget to each dataset (i.e., language), meaning that datasets of different sizes will be trained for a different number of epochs. Less common languages (with smaller datasets) will undergo more epochs compared to more widely spoken languages (which have larger datasets). The paper argues that this imbalance can lead to suboptimal model performance. The authors propose a highly intuitive and simple method for balancing the number of epochs across different datasets while staying within the overall budget B.

The method sets a maximum number of epochs N for any given dataset. The process starts with the smallest dataset (the least common language), determining the number of epochs E_i for that dataset based on its symbol count C_i and the average budget per language (computed as B/|D|). If the computed E_i exceeds N, it is capped at N. The budget allocated to this dataset is then subtracted from B, and a new average budget uiu_iui is recalculated for the remaining |D| - 1 datasets. This process is repeated iteratively for all datasets. The authors note that they do not use token count to estimate the "effective size" of each language due to the complexity of tokenization in multilingual datasets.

In the final step, all budgets uiu_iui are normalized using softmax to obtain the distribution ppp, from which data is sampled for each language. The distribution ppp can be adjusted using a temperature parameter τ (by raising p_i to the power of 1/τ), allowing for smoothing or sharpening of the distribution before sampling from it.

That's it for today—tomorrow, another paper on the topic…
https://arxiv.org/abs/2304.09151

Mike's Daily Paper – 29.03.25
Efficient Online Data Mixing For Language Model Pre-Training

Continuing my review of research on optimization of model training (particularly language models) when multiple datasets are available. Since I already defined the problem in my reviews from 26.03 and 28.03, I won't repeat it here and will jump straight into the core idea of this paper. This paper approaches the problem differently from the two previous papers I reviewed. However, despite some mathematical complexity, I find its proposed approach highly intuitive. The authors aim to solve the problem of constructing a training dataset D from multiple datasets D_1, ..., D_n using Multi-Armed Bandits (MAB).

For context, the MAB problem is defined as follows: We have n slot machines, each with an unknown probability of winning p_1, ..., p_n. The goal is to determine a selection strategy that maximizes total winnings (e.g., expected reward) over N trials Notice how our training optimization problem is quite similar to MAB - we need to determine the dataset selection strategy for training without knowing in advance the exact "impact" of each dataset on the final model performance.

Without diving too deep into the math (Markov decision processes, Gibbs distributions, etc.), the goal here is to find a probability distribution p_1, ..., p_n over the datasets to maximize model performance. The key twist is that this distribution evolves over iterations, where each iteration consists of a step (or a predefined number of steps) of training on data from the selected dataset D_i.

In other words:

- At each step, we sample a dataset based on the current **p** distribution.
- We train the model on data from that dataset.
- We update p based on the training results.

Naturally, this raises a crucial question: How do we determine p for the next iteration based on previous training results? This brings us to the concept of reward, which reflects the "success" of selecting dataset D_i at a given step. If training on D_i was successful, we increase its probability (at the expense of others), and if it was less successful, we decrease it.

So what exactly is the reward here? The reward is the information gain (IG)—essentially, how much the model benefits from the data in D_i. The paper estimates IG using perplexity (the exponent of the loss) on the dataset's data. This loss is estimated from a batch of

data.Additionally, the algorithm accounts for exploration—we don't want to drastically reduce a dataset's selection probability based on too few batches. To mitigate this, the paper applies a common technique from MAB and RL: adding a small ε□ value to every p_i, which decreases over iterations.

Final Algorithm: 3-Step Process

1️⃣ Update selection probabilities p_1, ..., p□.
2️⃣ Sample data from datasets D_1, ..., D□ based on these probabilities and train the model.
3️⃣ Update probabilities again based on the trained model's results from step 2.

A highly recommended paper—I really enjoyed diving into it!
Read the full paper here

Mike's Daily Paper – 01.04.25
OPTIMIZING PRETRAINING DATA MIXTURES WITH LLM-ESTIMATED UTILITY

The paper I will briefly review today is a generalization of the paper I reviewed on 28.03 about the UniMax method (though there are quite significant differences between them, and in my opinion, it is more similar to the paper I reviewed two days ago, on 30.03). This will be the last review (for now) in the series of papers I've covered on training optimization when multiple datasets D_1, ..., D_n are available.

The goal here, as in the last three reviews, is to propose a distribution w_1, ..., w_n for optimal sampling from these datasets to maximize the model's performance on a predefined validation dataset. This is done under one of two constraints:

- A token budget constraint (data-constrained training).
- A compute budget constraint (though in my opinion, each can essentially be formulated in terms of the other).

The paper formulates the training optimization problem in an original way that I haven't seen before. It is posed as an optimization problem (minimization) with constraints (note: there is a mistake in Equation 2- it should be argmax instead of argmin). The loss function consists of two terms: The first term is the norm of the difference between vector of ones (denoted as **1** in the paper and the inner product of the weight vector ww and a vector containing the performance scores (normalized to be between 0 and 1) of the model on benchmarks (several per dataset) for datasets D_1, ..., D_n (referred to as utility in the paper).

At first glance, this seems a bit strange, but the maximum value of this term is reached when the model's average performance on all benchmarks for all datasets is perfect (i.e., equal to 1, which likely represents maximum possible performance). The second term in the optimization objective is a regularization term **|D| |w^{T}w**. This term reaches its minimum value when the weight vector ww has equal values (i.e., w_i =1/n). In other words, the model is encouraged to assign equal sampling probability to all datasets - which makes sense, since we want the model to "see" as diverse a dataset mix as possible.

In addition, there are constraints on w: itt must be a valid probability distribution while another constraint limits the total token budget for the entire training process.

The paper uses the Splitting Conic Solver (SCS) to solve this optimization problem (seems non-trivial, but I didn't dive too deep into it). Of course, this approach requires computing the objective function described in the previous paragraph, which involves evaluating performance across n benchmarks. This is a computationally expensive process, and the authors suggest a method to reduce the computational effort required.

The authors propose a method to approximate this evaluation using language models. First, a powerful model (they used a large LLAMA) is asked to summarize each benchmark task based on its examples. Then the model also constructs a utility prediction prompt (for the trained model), designed to estimate the trained model's performance on the benchmark. The goal (as far as I understand, since this part was less clear to me) is to, given a small number of examples, estimate the model's performance on the benchmark (there are five possible scores).

And that's it- this concludes my series of reviews on training optimization when multiple datasets are available. See you in the next topic**s!**

🔗 **Read the paper**

Mike's Daily Paper – 02.04.25
SymDPO: Boosting In-Context Learning of Large Multimodal Models with Symbol Demonstration Direct Preference Optimization

Today, I'm making a sharp pivot in my review topics and covering a paper on training multimodal models (specifically, MLLMs). The paper proposes a method for training models on in-context learning tasks, where the model receives several demonstrations—each consisting of an image, a question, and an answer. The model is then expected to answer a question about a new image (which may contain the same characters, for example) based on the provided demonstrations. This review will be short and light.

The authors propose a way to improve the understanding of relationships between different modal data sources in multimodal models. For example, models that support both language and images sometimes struggle with tasks requiring semantic understanding between visual and textual data—such as the in-context learning task for MLLMs described above. The paper highlights that MLLMs often fail these tasks and sometimes answer questions without considering the provided context (i.e., the images, questions, and answers). The paper proposes a fine-tuning method for multimodal models to address such failures.

The proposed fine-tuning method is based on Reinforcement Learning with Human Feedback (RLHF), specifically using Direct Preference Optimization (DPO) - a modification of the popular Proximal Policy Optimization (PPO). The main advantage of DPO over PPO is that DPO does not require training a reward model. Instead, it only needs a dataset of question-answer pairs, including both preferred and non-preferred answers. The core idea in this paper is to engineer such a dataset for the given use case and apply DPO for multimodal model fine-tuning.

In essence, the paper suggests manipulating questions and answers in various ways to force the model to fully consider the given context. Some of the tricks include:

- Providing a nonsensical (random word) correct answer.
- Replacing an undesirable answer with gibberish.
- Removing the question entirely while keeping the answers unchanged.

There are a few more tricks like these, and by combining them, the paper achieves a better multimodal model using DPO fine-tuning.

As promised, a short and light review.

https://arxiv.org/abs/2411.11909

Mike's Daily Paper – 03.04.25
Amortizing intractable inference in diffusion models for vision, language, and control

Introduction:

Diffusion models have revolutionized generative AI, enabling stunning image synthesis, advanced text generation, and even decision-making in reinforcement learning. These models operate by gradually refining random noise into structured data, effectively learning a probabilistic prior over complex distributions.

However, many real-world applications require more than just unconditional generation—they demand posterior inference, where the model generates samples that satisfy specific

constraints. Posterior inference in diffusion models typically relies on classifier guidance, score-based reweighting, or fine-tuning methods like KL-regularized optimization. But these approaches suffer from major drawbacks:

Bias and mode collapse: Heuristic guidance (like classifier reweighting) distorts the true posterior.

Computational inefficiency: Sampling methods often require costly iterative refinements.

Lack of generality: Existing techniques work well for specific tasks but don't transfer across domains.

The authors of "Amortizing Intractable Inference in Diffusion Models for Vision, Language, and Control" propose a fundamentally different approach, inspired by reinforcement learning (RL) and Generative Flow Networks (GFlowNets). Their method, called Relative Trajectory Balance (RTB), reframes posterior inference as a sequential decision-making problem. This enables efficient, unbiased, and generalizable sampling from constrained distributions without relying on explicit likelihood reweighting.

Reframing Posterior Inference as a Trajectory-Based Learning Problem

At its core, posterior inference in diffusion models requires sampling from a conditional distribution:

$p(x|c) \propto p(x)f(x,c)$

where:$p(x)$ is the pretrained diffusion model (our prior over data). $f(x,c)$ is an auxiliary constraint function (e.g., a classifier, a language model, or a reward function). The goal is to generate samples x that are consistent with the constraint c, without distorting the learned prior too much. Traditional methods attempt to approximate $p(x|c)$ by modifying the diffusion process with heuristic guidance techniques, such as:

- Classifier guidance, which modifies the score function using gradients from an external classifier.
- Likelihood reweighting, which applies importance sampling to adjust sample probabilities post hoc.
- Direct optimization, where the diffusion model is fine-tuned using KL constraints.

The problem? These methods struggle with posterior accuracy, sample diversity, and computational efficiency.

Enter GFlowNets: A New Approach to Learning Probability Distributions

GFlowNets are a family of generative models that learn to sample from complex distributions by treating the generation process as a sequence of decisions. They are inspired by reinforcement learning, but instead of optimizing for a single best trajectory (like in traditional RL), GFlowNets learn to generate diverse samples proportionally to a given reward function.

In VAEs or GANs, we train a model to directly map latent variables to data samples. In diffusion models, we iteratively denoise a latent variable to obtain a final sample. GFlowNets take a different approach. They define the generative process as a trajectory through a state space (e.g., a sequence of denoising steps in a diffusion model). They assign a probability to each trajectory such that the likelihood of reaching a sample x is proportional to a given reward function R(x). They ensure balanced forward and backward transitions to achieve unbiased, amortized inference.

In simpler terms, GFlowNets turn sampling into a decision-making process, where the model actively chooses how to construct a sample rather than passively generating from a prior. This is precisely the missing link in posterior inference for diffusion models. Instead of relying on heuristic guidance, we can learn a policy that directly models the posterior distribution. Instead of requiring expensive iterative sampling, we can amortize inference by training the model to efficiently traverse the sampling space.

The RTB Approach: Learning Posterior Distributions via GFlowNet-Inspired Training

The authors introduce Relative Trajectory Balance (RTB), which moves away from heuristic guidance and instead learns the posterior distribution through reinforcement learning principles. The key insight is that sampling from a constrained posterior can be viewed as a sequential decision-making process, where each step in the diffusion model corresponds to a state transition in a Markov Decision Process (MDP).

Instead of passively sampling from a fixed prior, RTB learns a policy that selects the most probable inference trajectories. This policy determines the sequence of latent variables x_t in the diffusion process, balancing between prior consistency and constraint satisfaction. Inspired by GFlowNets, RTB learns a probability model over inference trajectories, ensuring that forward and backward sample generation are balanced. This prevents biasing the learned posterior while still allowing efficient constraint satisfaction. Unlike traditional methods that require computing explicit likelihood ratios, RTB directly learns the optimal trajectory policy through relative probability adjustments, making it computationally more efficient.

Why This Matters: A New Paradigm for Diffusion Model Inference

RTB represents a fundamental shift in how we approach posterior inference in generative models. Instead of manually tweaking diffusion guidance techniques, it learns the posterior distribution itself using reinforcement learning principles.

This approach offers several major advantages:

Unbiased posterior estimation: Unlike classifier guidance, RTB does not introduce distortions in the learned posterior.

Generalization across domains: The same framework can be applied to vision, language, and reinforcement learning.

Computational efficiency: By amortizing inference over training, RTB avoids costly iterative reweighting.

Mike's Daily Paper – 05.04.25
GIVT: Generative Infinite-Vocabulary Transformers

Today we're going back a few years to the time when terms like VAE, VQ-VAE, and VQ-GAN attracted the kind of attention that generative diffusion models receive today (though admittedly less than agents, but still). The paper we'll review today presents an interesting enhancement to VQ-VAE, which caught my attention since, as mentioned, papers in this area have become a "rare bird" in our (AI) landscape.

First, I'll give a short introduction to VQ-VAE. Let's start with VAE, which stands for **Variational AutoEncoder**, invented back in 2014 by the legendary Kingma. Basically, a VAE consists of two networks: an encoder and a decoder. The encoder produces a latent representation (or embedding) of a data sample, and the decoder turns this latent representation into an image. The encoder outputs the parameters of a Gaussian distribution (a mean vector and a diagonal covariance matrix), from which the latent vector is sampled and then passed to the decoder to reconstruct the input image.

The loss function of the VAE is based on **ELBO** (Evidence Lower Bound) and includes two components. The first is the reconstruction loss – in the classic VAE, it's the norm of the difference between the reconstructed and original image (in more advanced versions, perceptual loss and GAN-style losses are added). The second component is the **KL divergence** between the distribution of the latent representation derived from the data (represented by the mean vector and diagonal covariance matrix) and a standard normal distribution. During inference, we sample a latent vector from a standard normal distribution and feed it to the decoder.

A notable and very popular improvement to the VAE is **VQ-VAE**. Instead of defining the latent space as a Gaussian distribution, it is defined in a **discrete** manner. Each patch in the image is described (in latent space) by a vector from a **codebook** of finite size, which is trained together with the encoder and decoder. In other words, there is a **finite number of latent representations** for each patch (we'll come back to this in a moment). Once the encoder, decoder, and codebook vectors are trained, an additional model is trained to predict the latent representation of patches, using all the latent representations from the dataset. This model (say, a Transformer) is trained to **autoregressively predict** the codebook vector (i.e., its index) of the

next patch given the previously generated patches. Then, the latent vectors of the patches are fed into the decoder to generate data (images).

As mentioned, there's a limited and finite number of latent representations per patch, which **restricts the semantic richness** of the images that VQ-VAE and similar methods can generate. And that's exactly where the paper we're now reviewing brings innovation—it proposes a method to **switch to a continuous representation** (without a codebook) of the latent vectors. But how is this possible? Let's recall that in VQ-VAE, we predict a categorical distribution over the codebook—meaning the final layer of the autoregressive model is a softmax over the codebook size.

So, can a model be trained to generate **continuous latent representations** autoregressively? The answer is yes. In the first stage, the paper trains a **standard VAE** as described earlier. This is done at the patch level, meaning the latent representation of an image is composed of the representations of its patches. In the second stage, the authors train a **causal Transformer** that predicts the next token's representation by predicting parameters of a **Gaussian mixture distribution**, from which the latent vector itself is sampled. That is, each time the causal Transformer (which considers only already generated vectors) predicts the mean vectors, parameters of diagonal covariance matrices for each mixture component, and the mixture weights. After these latent vectors are predicted and sampled, they are fed into the decoder to generate an image.

It's worth noting that **GIVT**, unlike VQ-VAE, can be trained **end-to-end** with the encoder and decoder, which the authors argue might be problematic. As an alternative to the causal Transformer, the authors propose training a model called an **adapter** (based on **Normalized Flow**) to generate the entire latent representation of the data **after** the encoder and decoder have already been trained—thus separating the two stages.

Additionally, the paper suggests training a **non-causal Transformer** to predict latent patch representations, similar to **masked language modeling (MLM)**. This approach, which I wasn't previously aware of, was introduced in the paper **MaskGit**.

https://arxiv.org/pdf/2312.02116

Mike's DailyPaper – 07.04.25
JETFORMER: An Autoregressive Generative Model of Raw Images and Text

Today's review is a continuation (even though there's no VAE here) of my previous review from 05.04.25. The paper we'll look at today proposes a method for training a multimodal model where a single autoregressive model is trained for both modalities (images and text).

In most multimodal models, separate encoders are used for text and image. According to the authors, this could be problematic (and I kind of get it). So, the paper suggests training one autoregressive transformer for both modalities together.

So how does this work? The paper proposes using a pretrained Normalizing Flow (NF) model to build image representations. An NF model trains a reversible mapping—meaning lossless—from data space (the image) to a space with a simple distribution (say, standard Gaussian). This mapping is usually constructed by composing multiple simple functions (e.g., over subsets of dimensions), and all these mappings are trained jointly with the goal of maximizing the likelihood of the data under the transformation.

In practice, the authors train an NF for each patch in the image (a patch representation is called a visual token). So the authors jointly train the NF model (for image representation) with the autoregressive transformer to generate both images and text. That is, given an image and its description (the order of input matters!), the transformer is trained to predict the visual token representations after NF (which are trained jointly with the transformer).

When the image is given before its description, the transformer is trained to reconstruct the textual token representations. Just like in the previous review (on GIVT), the model predicts the parameters of a Gaussian Mixture for each token, and the actual representation is sampled from that distribution.

The paper also proposes improving the robustness of the representations generated by the trained autoregressive model using data noising (as I understand, only the visual data is noised). This is done in a curriculum-style approach. Initially, the data is strongly noised so that the model can learn the “coarse” aspects of the data. Over time, the noise is reduced, allowing the model to focus on learning the finer details.

https://arxiv.org/abs/2411.19722

Mike’s Daily Paper – 09.04.25
O1-CODER: An O1 Replication for Coding

I finally got around to reviewing this paper that caused quite a stir when it came out. The authors’ explicit goal is to replicate OpenAI’s O1 model, but focused specifically on coding tasks. The paper uses RLHF techniques combined with a self-play method where the model learns from data it generates itself. It starts from a dataset of coding questions and their corresponding answers (i.e., code 😊).

The core idea of the paper involves six main stages.
Stage 1: The authors build a tool (not described in much detail) that generates comprehensive tests for a given coding question and its correct solution. This tool (called TTG) will later be used to estimate the reward for code outputs produced by O1-CODER.

Stage 2: Using MCTS (Monte Carlo Tree Search), the model constructs reasoning chains for examples in the dataset. MCTS is a planning algorithm for decision-making that samples the state space (tokens, in our case) to estimate the reward of different possible actions. It incrementally builds a search tree, choosing at each step to expand the branch (token sequence) that looks most promising, balancing exploration of new paths with exploitation of known good ones. Each path (i.e., reasoning chain including a proposed solution) gets a reward of 1 or 0 from TTG (pass or fail all tests).

Stage 3: The model undergoes supervised fine-tuning (SFT) on reasoning chains that led to correct solutions (with reward 1).

Stage 4: Iterative self-play training begins. In each iteration, the training dataset is enriched with new examples generated by the model itself. The process involves either SFT on correct-answer chains (excluding iteration 0) or RLHF training using DPO (Direct Preference Optimization) on pairs of positive and negative examples.

Then, the model generates new reasoning chains (excluding the final answer), and a Process Reward Model (PRM) scores these partial solutions. The model then completes the answer from the reasoning chain, and new tests are generated for the question (since the correct answer is known — likely because the questions come from a benchmark dataset).

The reward is computed by running the tests on the model's generated code: 1 if all pass, 0 otherwise. This is then combined with the reasoning rewards via an aggregation function. The model is trained to maximize this combined reward using some RL method (possibly regularized, though the paper doesn't go into details).

Finally, new examples are generated using the updated model, added to the dataset, and Stage 4 is repeated (self-play loop).

Really interesting paper…https://arxiv.org/abs/2412.00154

Mike's Daily Paper – 11.04.25
Arithmetic Without Algorithms: Language Models Solve Math with a Bag of Heuristics

I've previously reviewed a few papers on using language models to compute arithmetic expressions involving standard mathematical operations like addition, multiplication, and so on. Personally, I think language models aren't naturally suited for these tasks (after all, we have calculators, Python, and such for that), but the research in this area is still fascinating. And there's another reason I chose this paper — it was written by Israeli researchers, and I always enjoy highlighting local work.

As the title suggests, the paper investigates what happens inside a transformer model when it's given an arithmetic task. Specifically, the authors attempt to locate the so-called computational circuit (or just circuit for brevity) within the transformer — that is, the components that actually perform the "necessary calculations" for such tasks. As you may recall, a transformer block consists of two main layers (plus normalization layers): a multi-head attention mechanism (MHA) and an MLP layer made up of two linear transformations with a nonlinearity in between. The computational circuit is composed of specific neurons in the MHA or MLP layers.

To identify this circuit, the authors use activation patching, replacing activations of certain neurons within the transformer to estimate the importance of each MLP layer and attention head at every input position (in this case, arithmetic prompts). How does it work? They take a specific arithmetic prompt (e.g., "226 − 68 =") and a random counterfactual prompt that leads to a different answer (e.g., "21 + 17 ="). After computing the model's activations for the counterfactual prompt, they feed the original prompt into the model.

At this point, they intervene in the computation (patching) by replacing the activation of a single MLP layer or attention head with the precomputed activation from the counterfactual prompt. Then they observe how this intervention affects the probabilities of the two possible answer tokens (the correct one and the counterfactual one). A specific formula is used to quantify the change in token probabilities. Once the circuits are identified, the authors evaluate their function by replacing all other activations in the model with average activations across a large dataset of arithmetic prompts — leaving only the computational path intact. They show that this replacement has almost no effect on the logits of the correct answer.

After identifying these circuits, the authors try to interpret their arithmetic meaning - and the result is quite interesting. It turns out these circuits operate as heuristics that help the model solve math problems. For example, some neurons estimate whether the result is in the range [150, 180], or whether it's divisible by 5. Combining these heuristics enables the model to solve relatively simple arithmetic tasks — especially those that don't involve very large numbers. This helps explain why LLMs often struggle with arithmetic over larger values.

There are a few additional insights: most of the critical components of the circuits are found in the MLP layers rather than in the attention heads. Another interesting finding is that the model tends to “converge” on the correct answer quite early — it can be extracted from intermediate layers using just a linear head.

https://arxiv.org/abs/2410.21272

Mike’s Daily Paper – 13.04.25
ONE STEP DIFFUSION VIA SHORTCUT MODELS

This paper presents an interesting approach to training generative diffusion models — an enhancement of the Flow Matching (FM) technique, which has become the leading method for training diffusion models. Essentially, the paper trains a model to estimate a trajectory (typically a straight line, which is the simplest path, though other works have used more complex ones) between a simple Gaussian distribution and the data distribution (images, video, or audio). The main claim is that using this method, one can generate data in just a single iteration.

The model is trained to generate a velocity (i.e., a gradient) along this trajectory at every time point t, where t represents the noise level which goes from pure noise (t = 0) to data point (t = 1). Once this velocity is estimated, a data sample can be generated by numerically solving an ODE, plugging in the predicted velocity along the way. For a linear trajectory, this velocity is constant (as it’s the derivative of a straight line). However, this linear assumption sometimes fails in practice as the resulting trajectories can be nonlinear and complex, which leads to poor sample quality.

To improve this, the paper suggests replacing straight-line paths with piecewise linear trajectories, essentially a type of linear spline rather than forcing the model to always follow a global straight path. The displacement of a data point within each small segment depends only on the current point x_t, the timestep t, and the spline granularity d (I'll expand on that later). These short segments are referred to as shortcuts in the paper. The model is trained to estimate them using a consistency loss, which encourages the model to behave “consistently” across consecutive shortcut segments. This loss is derived from a simple combination of the update rules for two adjacent shortcut steps.

Then, the authors combine this consistency loss with the standard FM loss (computed along the straight trajectory). Since the shortcut path can be constructed with different granularities, i.e., varying numbers of linear sub-segments; the training leverages this by training the model across

multiple resolutions. Given a timestep t (noise level), a noisy input sample, and a spline granularity d, the model is trained to predict the data point shift over the next segment (and there are d such steps total). Then, an ODE is solved to move the point accordingly. That new point is again passed through the model to predict the next shift. The consistency loss is then applied across two such consecutive shifts.

Overall, a really interesting and well-written paper, I highly recommend reading it!

https://arxiv.org/abs/2410.12557

Mike's Daily Paper – 14.04.25
Draft Model Knows When to Stop: A Self-Verification Length Policy for Speculative Decoding

This paper caught my eye at first glance because of the phrase *"Speculative Decoding"* or SD for short — something very close to my heart. I even prepared a fairly comprehensive presentation on it, which I present in various forums. SD allows increasing the text generation rate of a language model by combining the target model with a smaller, faster (and of course weaker) model. The small model autoregressively generates a few tokens, and the target model then uses these tokens to predict its next tokens *in parallel*. This can significantly boost the sampling rate of the large model.

The method leverages the fact that the bottleneck in the generation process is data transfer between the different types of GPU memory (in particular, the large and slow HBM vs. the fast SRAM located close to the compute units). SD performs a fast prediction with the small model, followed by parallel prediction with the large model, using the tokens generated by the small model.

But there's a catch, of course: in order to sample from the same token distribution as the large model (while using the small model's predictions), we need to perform something akin to rejection sampling (RS).

To recap: RS allows sampling from an easy-to-sample distribution fff in order to produce samples from a different, harder-to-sample distribution ggg. We sample a point xxx from fff, and accept it with probability equal to the ratio g(x) / f(x) (if the ratio is greater than 1, the point is always accepted). One can prove that the accepted samples follow the desired distribution ggg.

So in our SD case, we do something similar for tokens generated by the small model. In the second phase (parallel sampling with the large model), for each token produced by the small model, we compute the ratio of the probabilities assigned by both models and accept the small model's token with probability equal to that ratio. As soon as the first token is rejected, the large model resumes generating from that point, and the small model is used again to produce the next speculative tokens. By the way, even the accepted tokens are reweighted — they're generated based on a mixture of the small and large model distributions.

As you might've guessed, *controlling the acceptance rate* of the small model's tokens is very important — ideally, we'd only generate tokens that will be accepted. This paper proposes a method to improve the acceptance rate.

It shows that the average acceptance rate (pretty straightforward) equals 1 minus the total variation distance (TVD) between the conditional distributions of the two models (given context). Luckily, we have access to a not-so-well-known inequality that provides a lower bound on the TVD using the difference between the cross-entropy of the two models' distributions (for a given token given context) and the entropy of the small model.

But of course, we can't compute the cross-entropy between these distributions during the large model sampling phase — because we sample all tokens simultaneously and don't know the conditional distribution for each token in advance. So, the paper estimates this cross-entropy using a time-averaged estimate: a constant (slightly greater than 1) times the entropy of the small model's token. Once we have this cross-entropy, we can estimate the acceptance rate for each small-model token *before* sampling with the large model. This allows us to set the number of speculative tokens to generate — we just keep generating until the estimated acceptance rate drops below a certain threshold.

Nice idea overall, but I think the choice of the constant in the final step isn't optimal, and I hope to see future work that improves this aspect of the method.

https://arxiv.org/abs/2411.18462

Mike's Daily Paper: 15.04.25
Classifier-Free Guidance inside the Attraction Basin May Cause Memorization

Back to diffusion models! This time, we're looking at a relatively lightweight paper (at least compared to the average diffusion model paper). It proposes a method for preventing memorization by diffusion models. Memorization here can be viewed as a form of mode collapse (throwback to the GAN era), where the model generates very similar images (often resembling the training set) from different latent inputs, usually sampled from a simple distribution like a standard Gaussian.

This phenomenon tends to appear most often in conditional diffusion models — the ones that generate images based on a text description (i.e., a prompt). In these cases, memorization occurs when, no matter what Gaussian noise you start with, the model ends up generating nearly identical images. This paper digs into the root cause of this behavior and concludes that it stems from the use of a popular technique called Classifier-Free Guidance (CFG).

CFG is intended to "pull" the generated image closer to the semantic meaning of the prompt — in other words, to make the image more aligned with the given description. As you probably

know, diffusion models generate images by gradually denoising from pure noise (typically Gaussian), in an iterative process. In each iteration, the model predicts the noise to be removed, given a noisy image and the timestep *t* (which is also part of the model's input).

CFG works by nudging the image toward the prompt using a weighted difference between two noise predictions: one from a conditional model (with the prompt) and one from an unconditional model (no prompt). This shifts the generated image away from the “average” unconditional result, and brings it semantically closer to the prompt. However, as the authors point out, CFG can pull the image too strongly toward the prompt — which can lead to memorization. Interestingly, they found that if you delay the use of CFG until later iterations (i.e., after some noise has already been removed), the memorization effect almost disappears.

Why? Because the norm of the conditional noise vector is much larger than that of the unconditional one in early denoising steps, but the two norms start to converge around the middle of the backward process.

So what’s the proposed solution?Run early denoising without CFG and then activate CFG in later iterations — once the conditional and unconditional noise norms begin to align. How do you know when to switch on CFG? Simple: when the gap between the two noise norms starts shrinking. That’s the core idea of the paper.

There’s also a solid set of mathematical definitions around what constitutes memorization (which I personally love), but if you’re not into the math, this summary should give you a good high-level understanding.

Paper link: https://arxiv.org/abs/2411.16738

https://arxiv.org/abs/2411.16738

Mike’s Daily Paper: 17.04.25
Memorization to Generalization: The Emergence of Diffusion Models from Associative Memory

Okay, continuing with a deep theoretical Paper on generative diffusion models. The central contribution of this paper is a precise reinterpretation of diffusion models as stochastic, overparameterized associative memory systems, particularly those that share structure with modern Hopfield networks. The core insight is that the behavior of diffusion models across varying data regimes — specifically the transition from memorization to generalization — mirrors the dynamics of Hopfield-type networks as they move past their critical memory capacity.

In classical associative memory models like Hopfield networks, the energy landscape is shaped by the patterns stored in memory. When the number of stored patterns is below the theoretical storage limit, each memory corresponds to a stable fixed point — an attractor — in this energy surface. As the number of stored patterns approaches the capacity limit, interference between patterns leads to the emergence of spurious attractors. These are stable points not corresponding to any training pattern, but often lying close to linear combinations or interpolations of the stored set.

This paper observes that diffusion models — which learn a time-indexed score function for denoising — implicitly define an energy functional over data space at each time step. Specifically, the score function learned during training can be interpreted as the gradient of the negative log-probability of the data at that time step, which aligns with the gradient of an energy-based model. When written out, this energy turns out to have the same structure as that of modern Hopfield networks with softmax-based energy functions, including a similarity-based aggregation over the training samples. In other words, diffusion models define a learned energy landscape that attracts samples toward training data — and potentially toward spurious attractors when the model cannot exactly memorize due to data overload.

What makes the connection non-trivial is that this structure is not imposed manually — it *emerges* from the training objective of the diffusion model. The denoising score-matching loss, which trains the model to reverse a stochastic corruption process, ends up instantiating an energy surface with attractor-like properties.

Now, as the training dataset size increases, the nature of the learned attractors changes. In the small-data regime, the score function learned by the diffusion model has high precision and confidence near training examples, and the generated outputs tend to be direct or near-exact copies — evidence of pure memorization. The model's score landscape is steep and concentrated, and the attractor basins are narrow and centered directly on the training samples.

As the training data size increases, the model can no longer dedicate a distinct, isolated basin to each sample. The energy surface begins to interpolate. Spurious attractors emerge — regions in latent space that do not correspond to any training example, but still act as local minima. These spurious regions reflect structural overlap between training patterns. Unlike memorized samples, spurious samples are supported by the synthetic distribution learned by the model but do not occur in the training data themselves.

The authors quantify this with three theoretical constructs:

- **Memorization capacity**: the maximum training set size for which the model primarily reproduces training data with high probability.
- **Spurious capacity**: the size at which the model most frequently produces samples that lie outside the training set but still within the model's learned distribution — these are not fully generalized yet.
- **Generalization capacity**: the minimum training set size after which the model begins producing samples that are dissimilar to both the training and previously generated samples — this marks true generalization.

The transition between these regimes is not smooth but exhibits phase-transition-like behavior. Empirically, the fraction of memorized samples drops sharply, the fraction of spurious samples peaks, and then drops as generalization takes over. These transitions are detected using nearest-neighbor based metrics that measure whether a generated sample matches training data (memorization), matches synthetic but not training data (spurious), or matches neither (generalized).

Crucially, the model does not simply interpolate between samples due to overfitting noise — the interpolation is a function of structural interactions in the score landscape. The spurious attractors are formed by the aggregation of contributions from many training samples under the softmax dynamics of the score function. This is functionally identical to how modern Hopfield networks form spurious attractors when the overlap between stored patterns increases beyond a certain threshold.

Thus, the key contribution is a rigorous theoretical bridge: **diffusion models operate as associative memory systems that exhibit a well-defined phase transition from pattern recall to generative abstraction**. This transition is mediated by the collapse and recombination of energy attractors in the score space, governed by data distribution complexity, training set size, and model capacity.

The work doesn't merely draw analogy — it aligns the functional components of denoising score matching, softmax aggregation, and attractor dynamics into a cohesive interpretation. It frames generalization in diffusion models not as emergent from architectural inductive bias alone, but as a result of distributed attractor interference — a direct consequence of overloading the energy surface defined implicitly by the training dynamics.

https://openreview.net/forum?id=zVMMaVy2BY

Mike's Daily Paper: 19.04.25
Critical Tokens Matter: Token-Level Contrastive Estimation Enhances LLM's Reasoning Capability

A fairly interesting paper focused on improving the reasoning capabilities of language models for questions that have clear, deterministic answers—such as math problems or coding tasks where correctness can be verified through a comprehensive test suite.

The authors introduce the concept of a critical token—a token that effectively serves as a signal for whether the model is likely to produce a correct or incorrect answer to a given question.

They observed that within incorrect reasoning paths, certain tokens almost always lead to wrong answers. These tokens tend to disrupt logical flow, distort dependencies, or introduce computational errors, thereby significantly affecting the final result. Unlike other tokens that may have a negligible effect on inference, these critical tokens act as failure points. Identifying them is crucial: avoiding or correcting them can often lead to a correct result—even along an otherwise incorrect reasoning path.

The paper proposes a method for identifying such tokens. A token is deemed critical if all reasoning paths that begin with it result in incorrect answers, and for all following tokens, 95% of the paths that start from them also end up incorrect. Note that some tokens may appear later in the sequence, after the critical token, and not all of their reasoning paths include that critical token—so it's entirely possible that some of these later tokens still lead to correct answers.

The authors conduct several checks to ensure that the tokens they've identified are truly critical in practice.

Next, the paper develops an RLHF-style alignment method, centered around minimizing the likelihood of generating critical tokens (since they are associated with errors). To do this, they fine-tune two separate models: one that generates correct answers, and another that intentionally generates incorrect answers (yes, you read that correctly).

They then formulate a method for estimating how likely a token is to be critical, given a prompt and the preceding answer tokens. This is done via a formula that computes the difference in

weighted likelihoods (conditioned) of the token between the correct-answer model and the incorrect-answer model. This estimate is low for tokens appearing in correct answers, and high for tokens likely to appear in incorrect ones.

In the final stage, they fine-tune the model using DPO (Direct Preference Optimization) over pairs of prompts with correct and incorrect responses. To reduce the chance of critical token generation, they modify the DPO loss term associated with the incorrect answer: they multiply it by the negative likelihood that a token is critical. Since this adjustment operates at the token level, the DPO process in this work is token-level, rather than sample-level as in the original formulation.

https://arxiv.org/abs/2411.19943

Mike's Daily Paper: 21.04.25
Normalizing Flows are Capable Generative Models

I chose this paper for review because it includes a collection of methods and approaches that are rare to encounter in today's deep learning literature. The second reason is the presence of a generative method called Normalizing Flows (NF for short). Like GANs and VAEs, this method lost—by knockout—to diffusion models in the battle for dominance in generative modeling. Still, it's a very interesting approach with a precise mathematical formulation and an intuitive core.

So, the authors are trying to bring it back into relevance, proposing an NF-based approach augmented with a few mathematical tools from the diffusion models world, plus a neat trick known as Tweedie's formula.

Let's start with: what is NF? NF is a training method for generative models that learns a one-to-one mapping between a simple distribution (like a standard Gaussian) and the data distribution (say, a dataset of images). Since the mapping is bijective, it's invertible, meaning that for any given data point we can easily compute its likelihood under the NF model — just apply the inverse mapping and evaluate the resulting point under the simple distribution. Naturally, the dimension of the latent space (defined by the simple distribution) matches the data space.

Most NF models are built as a composition of simple, bijective mappings that act on subsets of the data dimensions (the rest are left unchanged). For images, for example, each such atomic transformation acts on a small set of pixels. Typically, these mappings are constructed from learned upper-triangular matrices (with nonlinear functions), as these matrices are invertible and their inverses are easy to compute — a big advantage when calculating data likelihoods. Additionally, each such matrix is structured as a residual connection, i.e., it's a sum of the original values and their transformed counterparts.

The paper builds an NF model using this structure, but also proposes several additions. The first is training the model on noised data — that is, adding a bit of Gaussian noise to the training

data. According to the authors, this increases the model's robustness (which makes intuitive sense). But to ensure that the model doesn't generate noisy samples as a result, they apply Tweedie's formula to denoise the output. This formula estimates the *expectation* of clean data given noisy observations, using samples from the noisy distribution and the gradient of the log-probability of the noisy data, scaled by its variance. That's how they get clean samples, even after training on noised data.

The last thing the paper introduces is Classifier Guidance, a well-known technique in diffusion-based generative models. Classifier guidance steers the sampling process using an external classifier. Instead of sampling based solely on noise, the model incorporates the gradient of the classifier's probability for a desired label — increasing the chance that the final sample belongs to a specific class. It's also possible to do this without a classifier — in that case, the sampling is nudged in the opposite direction of the unconditional model's predictions. The authors found a pretty intuitive way to inject this idea into the NF training: essentially shifting the sample after each NF transformation step.

A fun, though not trivial, paper — especially if you try to dive into the math. But hopefully I've captured the core ideas clearly.

https://arxiv.org/abs/2412.06329

Mike's Daily Paper: 23.04.25
The Broader Spectrum of In-Context Learning

Rethinking In-Context Learning: From a Narrow Definition to a Broad Capability

The authors suggest a major shift in how we think about in-context learning (ICL). Instead of seeing it as just the model's ability to learn from a few examples (few-shot learning), they define it as a broader ability to adapt to context — something that naturally emerges during pretraining on structured data.

At the core of this idea is a simple rule: if a model gets better at predicting the next word because of earlier words in the same sequence, that's ICL. This applies not just to few-shot learning, but also to things like coreference resolution (e.g., figuring out who "she" refers to), grammar rules, or staying on topic.

The authors describe two layers of this learning:

- **Outer loop**: During training, the model sees sequences that often have patterns (like question-answer pairs, lists, stories, or instructions). These patterns define "tasks" the model learns from.

- **Inner loop**: At inference time, the model adjusts its behavior based on the current context, using just its internal activations — no gradient updates needed.

This view shows that ICL isn't just about copying examples. It's about using clues in the context to understand what kind of task is happening and how to act accordingly — similar to how meta-learning works in other domains.

Types of In-Context Learning Beyond the Usual Few-Shot Setup

Many model behaviors that seemed like tricks actually turn out to be examples of ICL. The authors identify six key forms:

Instructional ICL

Instead of showing examples, you just tell the model what to do ("Translate English to French"), and it figures it out. The model learns to turn plain language instructions into behavior, even for new tasks.

Role-based ICL

If you tell the model, "You are a brilliant French translator," it changes how it responds. The model picks up on these role hints based on patterns it learned from text with different personas (e.g., Wikipedia bios, stories, conversations).

Explanation-augmented ICL

Adding explanations after examples helps the model improve — especially when questions are hard. Explanations help the model focus on the important structure of the task, not just surface patterns.

2.4 Unsupervised ICL

Even when the model sees input examples with no answers, it sometimes still improves. That's because it recognizes familiar formats (like quiz questions or math problems) and guesses what kind of task it's seeing.

2.5 Time Series ICL

The model can pick up patterns in time series (like sequences of numbers), even when no examples are labeled. It can combine trends like cycles and linear growth, just from observing the sequence. This shows the model can learn underlying dynamics from context.

2.6 Meta-ICL

When the model sees several different tasks in a row (each with its own examples), it gets better at later ones. This means it's not just learning tasks — it's learning how to learn tasks, using layers of internal representation.

3. How ICL Emerges from Natural Language Patterns

The authors argue that ICL develops naturally from the types of patterns found in language — not from specific labels or training tricks.

3.1 Coreference Resolution

To figure out who "she" refers to in a sentence like "Alice lifts weights. She goes to the gym," the model must track identity and gender across sentences. Harder cases (like Winograd examples) require logic and real-world knowledge.

3.2 Parallel Structure

When the model sees repeated formats (e.g., "Alex likes cats. Jordan likes dogs."), it can generalize a pattern or rule. This is similar to few-shot learning, but it happens in natural text.

3.3 Word-Sense Disambiguation

For a word like "bank," the model uses context to decide if it refers to a river or a financial institution. This is a kind of light classification using only context clues.

3.4 Syntactic Agreement

Even older models could learn to match verbs with subjects ("The dogs run", not "The dogs runs"). This is another example of ICL — learning grammar rules from context.

3.5 Topic Modeling

In long texts, the model shifts its vocabulary based on the topic — even if the topic isn't explicitly stated. This shows that it tracks topic continuity using context, though this form of ICL is more implicit.

4. How ICL Generalizes in Different Ways

The authors identify three key ways to test how well a model generalizes its in-context learning:

4.1 Generalization in What Is Learned

Can the model learn something truly new from the context, like a novel rule or category it hasn't seen before? This is the difference between retrieving known patterns and doing actual meta-learning.

4.2 Generalization Across Formats

Can the model learn the same task if it's shown in different forms — as examples, instructions, diagrams, code, or analogies? This tests how abstract and flexible the model's internal representations are.

4.3 Generalization in How It's Used

Can the model apply what it learned to a new domain (like using a number-based rule on words)? Can it explain the rule or write it as code? This pushes toward symbolic reasoning — using abstract understanding, not just surface patterns.

Conclusion

This paper challenges the narrow view of ICL as just few-shot function learning. Instead, the authors present ICL as a broad, flexible behavior that naturally emerges from language model training. It includes many kinds of learning from context — from role conditioning to task inference, topic tracking, and more. By treating ICL as a spectrum of learned behaviors, not a single mechanism, the paper lays the foundation for better benchmarks, deeper evaluations, and a more complete theory of how models generalize. It doesn't just expand the definition of ICL — it redefines it.

https://arxiv.org/abs/2412.03782

Mike's Daily Paper: 26.04.25
Multimodal Latent Language Modeling with Next-Token Diffusion

Today is Saturday, and today's review will be lighter and fairly short. The focus is on multimodal generative models capable of "understanding" and generating data from multiple modalities — meaning text, images, audio, and similar types. The paper essentially connects latent generative models for textual data with those for more continuous data (even though that data, too, is discretized). The authors achieve this by training generative diffusion models for different types of data in the latent space. In other words, the model is trained to generate latent representations both for textual data and for other types of data, like images and audio.

Unlike many other works, the authors do not only train the multimodal generative model itself but also train an embedding model responsible for producing latent representations for each modality. Typically, in diffusion models, the embedding model is based on a VAE (Variational Autoencoder). However, the authors propose a slight but important modification to the standard VAE: instead of having the encoder generate both a mean vector and a variance vector for the latent representation, it generates only the mean vector. The variances are then sampled from a fixed Gaussian distribution with a predefined variance (a hyperparameter). According to the authors, this change prevents the collapse (i.e., zeroing out) of the variance vector generated by the encoder, a problem that would otherwise harm the diversity of the generated images.

The training process extends beyond just text. The authors also train a VAE for non-textual data such as images and audio. These are divided into tokens — image patches for images and time segments for audio — and fed to the model as sequential data. It's important to note that the model treats data from every modality as sequential data. For text and audio, this is very natural since they have a clear inherent order. For images, although there is no strict natural sequence, one can still impose different orders (for example, left-to-right and top-to-bottom, or even right-to-left and bottom-to-top). This flexibility in sequencing image patches is an interesting design decision.

The diffusion model for non-textual data is trained to denoise a noisy latent representation given its context (all models in the paper are, of course, autoregressive). After the denoising step, the clean latent vector is fed into the VAE decoder in order to reconstruct the original data, with the overall training objective being to reconstruct the data as accurately as possible. For textual data, the noise is applied directly to the token embeddings, and a diffusion model is trained to recover them. Moreover, for textual data, a separate linear layer is trained to map the latent vector back into the token space (essentially, producing a softmax distribution over the vocabulary). It's worth noting that the diffusion models are trained jointly with the VAE components — both encoder and decoder — which creates a tightly coupled and efficient training procedure.

Finally, to clearly separate textual from non-textual data during generation, the authors introduce special tokens that act as separators between different modalities. This allows the model to better understand the modality it is working with at each point during generation.

https://arxiv.org/abs/2412.08635

Mike's Daily Paper: 28.04.25
Around the World in 80 Timesteps: A Generative Approach to Global Visual Geolocation

Today's paper review covers something a little different — and honestly, refreshing. It's not every day (or even every month) that I get to dive into a paper that applies machine learning models to geographic tasks. Yes, you read that correctly: the powerful deep learning tools we've developed over the last decade are finding their way into geolocation problems too.

The task tackled here is deceptively simple: given an image, predict the location on Earth where it was taken. Formally, for a given input image, the model must output the corresponding Earth coordinates. Since the Earth is a sphere (or at least a good enough approximation for machine learning purposes), the location can be conveniently represented as a point on a sphere, often parameterized by two numbers.

The authors approach this as a generative modeling problem. Specifically, they train a **diffusion model** that takes an image as input and produces the geographic coordinates as output. Now, remember how diffusion models work: they are trained to gradually denoise data — meaning that, at each step, given a slightly noisier version of a sample and the current timestep, the model predicts the noise that must be subtracted to move back to a cleaner version of the sample. Over the course of many such steps, starting from pure noise, the model progressively reconstructs clean data. This basic principle is what powers many state-of-the-art generative models today.

However, the landscape has evolved. Recent diffusion models are increasingly based on an idea called **Flow Matching (FM)**. In Flow Matching, instead of modeling the score (the gradient of the log-density), the model directly learns a **velocity field** — a time-dependent function that tells you how to move a point over time from the noise distribution toward the data distribution. Essentially, you supervise the model to predict the velocity required to transport noisy samples back to clean data along an idealized path. Some models keep this velocity constant across time; others, like the one in this paper, allow it to vary with the timestep ttt. Training becomes a regression problem: predict the correct velocity given a noisy sample and a time t. Once trained, sampling (inference) is performed by solving an **ordinary differential equation (ODE)** whose dynamics are governed by the learned velocity.

Now, if you're working with pixel data (where each pixel is a number between, say, −1 and 1), it's standard to define your noise as Gaussian. But in this paper's setup, the data isn't living in flat Euclidean space — it's living on the surface of a sphere. And if you want to stay faithful to the geometry, you can't just add Gaussian noise arbitrarily: you have to ensure that your noisy samples remain on the sphere at every timestep. This requirement brings us into the domain of **Riemannian geometry**. Instead of simple additive noise, the model perturbs data by **rotating** it - that is, moving it along the tangent space of the sphere, which represents all the directions you can locally travel without "falling off" the sphere.

This geometric constraint fundamentally changes how noise is applied and how the velocity field is defined. The simple derivative of x_t with respect to t used in Euclidean settings no longer suffices; instead, the velocity must be defined using operations that respect the spherical manifold structure. Additionally, when starting the inference process, the initial "pure noise" samples are drawn **uniformly** from the surface of the sphere - which, while conceptually simple, is actually not entirely trivial to implement correctly from a mathematical standpoint.

Despite all these adaptations, the core learning goal remains unchanged: the model is trained to estimate the correct velocity at each timestep t, given a noisy geographic coordinate associated

with a given image. The image itself is processed through a **frozen encoder** — meaning an encoder whose weights are not updated during the diffusion training — and the resulting embedding conditions the diffusion model throughout generation.

Overall, this is a very cool and thought-provoking paper. It's a great reminder that when generative modeling meets non-Euclidean geometry, new challenges - and opportunities - emerge. Anyone comfortable with (or interested in) Riemannian geometry will find a lot to enjoy in the details. Highly recommended!

https://arxiv.org/pdf/2412.06781

Mike's Daily Paper – April 30, 2025
THE COMPLEXITY DYNAMICS OF GROKKING

This is one of the strongest and most profound papers I've read recently. And no — it didn't train a model that achieved state-of-the-art results across benchmarks, nor did it propose a new architecture or training method. What the authors set out to do is explain a phenomenon called grokking through the lens of compression; both compression of data and compression of models.

This topic is a bit dense, so I'll try to explain it step-by-step in a simplified way.

What Is Grokking?

Grokking is a phenomenon that happens during neural network training when we continue training well after reaching the validation loss minimum. At first, as expected, the model enters an overfitting phase, and validation loss increases. But then something strange happens: At a certain point, validation loss begins to decrease again, suggesting the model is transitioning from memorization to generalization. In simple terms, the model actually "gets" the problem.

This typically occurs in overparameterized models, where the number of trainable parameters is much larger than what is theoretically needed to fit the dataset (There's a more precise mathematical explanation, but it involves nontrivial complexity theory and isn't needed for this overview). Grokking is related to the double descent phenomenon and also to the lottery ticket hypothesis line of research. Interestingly, if you continue training, the validation loss keeps dropping, it doesn't stop, meaning the loss converges toward zero.

So What Does This Have to Do With Compression?

To explain this, we need to introduce two key concepts: The first is called the Minimum Description Length (MDL) principle. This principle says that if we want to optimally compress a dataset using a model, we need to minimize the sum of two terms:

- The entropy of the dataset *after being passed through the model*, and
- The complexity of the model itself.

This idea builds directly on Shannon's coding theorem, which says that the **l**ower the entropy of the data, the more effectively it can be compressed. Now, estimating the entropy of a dataset given a model is something we roughly know how to do. For classification tasks, for instance, you can simply use cross-entropy loss.

But estimating the complexity of a model is harder. We first need to define something called Kolmogorov Complexity (KC). The KC of a data string ddd is the length of the shortest computer program that outputs d. For example, a line of repeated 1s has low KC (it's easy to describe), while a random sequence of 0s and 1s has high KC (it takes nearly as much space to describe as the data itself). Importantly, KC is not computable in general.

Enter Rate–Distortion Theory

Another important concept is the rate–distortion function r, which given some input x and distortion tolerance ε defines the minimum number of bits (or KC) required to describe some output y that is within ε distortion of x. Of course, the notion of "distortion" depends on the chosen distance metric. In the paper's context x and y are models (represented by their weights) while x is a fully(regularly) trained model M and is: a coarse-grained version of M, denoted by CS.

A coarse-grained model is a simplified version of M, for example, through: Quantization, Pruning or Replacing weight matrices with low-rank approximations. Even regularized models can be seen as coarse-grained relative to an unregularized one. The distance function d(x,y) the authors use to compute r is the difference in loss between the full model and the coarse-grained one.

Back to Grokking

The paper's main claim is that as training progresses, the model becomes increasingly compressible. That is: there exists a coarse-grained model CS that achieves nearly the same performance (within ε) as the original model M. And this happens precisely during the grokking phase. Moreover, at this point, the description length of the data (as compressed through the model) starts declining steadily — meaning the model is now encoding the dataset using a simpler, more compressible structure.

Why is this meaningful? Because the model is achieving low cross-entropy loss through compression. Its rate–distortion value is dropping — it's finding simpler explanations for the same data.

Hope I managed to explain this paper clearly — it's one of those rare cases where theory, information compression, and deep learning dynamics come together to shed light on something truly counterintuitive.

https://arxiv.org/abs/2412.09810

Mike's Daily Paper – 02.05.25
On Speeding Up Language Model Evaluation

This paper tackles one of the most practical — yet rarely discussed — challenges in working with LLMs: how do you efficiently evaluate the performance of dozens or hundreds of prompts or models over large question sets without burning absurd amounts of compute?

Each evaluation run means invoking a heavy model — often with tens or hundreds of billions of parameters — on every example, for every prompt. When you're looking at hundreds of prompts and thousands of examples, that's hundreds of thousands of forward passes. And it's not even training — it's just evaluation, which somehow makes the inefficiency feel even more frustrating: we just want to know who's best, without paying the cost of running everything on everything.

The paper introduces two clever, adaptive algorithms that aim to solve exactly this.

The first, **UCB-E**, is based on the well-known **Upper Confidence Bound (UCB)** principle from multi-armed bandits. Instead of naively testing every method on every example, the algorithm builds an estimate of how good each method (say, model + prompt) is based on what's been tested so far, and adds an "uncertainty bonus" — just like in Monte Carlo Tree Search — to encourage exploring under-tested options. That way, it doesn't just exploit the top candidate, it also continues probing others that could surprise. Over time, it focuses evaluation budget only on methods worth knowing more about.

But the real innovation comes from the second algorithm: **UCB-E-LRF**.

Here, the authors make a much deeper observation: while the score matrix (methods × examples) might look like a huge, unstructured grid, in practice it has **latent structure**. Some examples are quite similar to others. Some methods behave in highly correlated ways. In other words, the matrix is approximately **low-rank** — it can be well-approximated using only a small subset of its entries.

The algorithm leverages this by starting with a small sample of real scores (say, 5% of the matrix), then training a **low-rank model**: each method and each example gets a vector embedding, and their dot product predicts the expected score. This lets the system infer the

entire rest of the matrix without explicitly computing it — just like classic recommendation systems that used low-rank matrix factorization. Better yet, it also estimates the uncertainty of each prediction. In each round, it decides where to spend the next evaluation: either where the uncertainty is highest, or where it might discover the best method. The system gradually learns the true structure of the problem, and focuses compute exactly where it could impact the final ranking.

This approach makes smart use of structure in the data, generalizes beyond observed scores, and remains fully **adaptive** — it doesn't assume upfront which method will win. Most importantly, it can save **85–95%** of the runs you'd need in a naïve setup. In LLM terms, that's the difference between a system you can run on a local GPU vs. one that costs thousands in cloud compute.

I was impressed by the combination of decision-theoretic tools (like UCB) and modern matrix approximation techniques (like factorization). It's a great example of how far you can go when you connect ideas across fields.

Highly recommended!
https://arxiv.org/abs/2407.06172

Mike's Daily Paper: 04.05.25
Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs

The paper presents a first-of-its-kind study focusing on a new phenomenon observed in advanced LLMs, termed "o1-like models" (such as OpenAI o1 and similar models). The core novelty of the paper lies in the identification, characterization, and proposal of solutions for the problem of "Overthinking" in these models.

To the best of my recollection, this paper (published at the end of December 2024) is the first to comprehensively define and analyze the "Overthinking" phenomenon in o1-like models. The phenomenon manifests as these models tending to allocate substantial computational resources (sometimes reflected in generating unnecessary tokens) even for very simple problems (like "2+3"). This involves generating long "Chains-of-Thought" (CoT) and numerous alternative solutions, often without improving the accuracy of the final answer.

The research empirically shows that later solutions in the chain-of-thought contribute very little to accuracy improvement (often, the correct answer appears in the first solution) and do not exhibit significant diversity in reasoning approaches (many solutions repeat the same approach). This phenomenon is particularly prominent in easy problems.

The authors define new efficiency metrics: The paper introduces two new efficiency metrics designed to quantify the rational use of computational resources by o1-like models, beyond conventional accuracy metrics:

- Outcome Efficiency: Measures the ratio between the minimum number of tokens required to reach the first correct answer and the total number of tokens generated. A low value indicates overthinking in terms of contribution to accuracy.
- Process Efficiency (ξP): Measures the ratio between the number of tokens contributing to solution diversity (i.e., tokens belonging to solutions presenting a new approach) and the total number of tokens generated. A low value indicates repetitiveness and lack of diversity in solutions.

The paper explores innovative strategies to reduce overthinking, based on the Self-training paradigm and Preference Optimization techniques, without needing external information. The novelty lies in applying these methods to the specific problem of simplifying responses while preserving inference capabilities. The authors used techniques such as SFT, DPO, RPO, and SimPO to train the model to prefer shorter, more efficient responses (identified as such from multiple samples), using the longest response as a negative example (found to be more effective than the default response).

The authors developed several new methods for creating a more efficient training dataset by deliberately truncating long responses:

- First-Correct Solutions (FCS): Retains only the minimal part of the response up to the appearance of the first correct answer.
- FCS + Reflection (FCS+Reflection): Extends FCS to also include the second solution that reached the correct answer, aiming to preserve "long-thinking" capability efficiently.
- Greedily Diverse Solutions (GDS): A greedy expansion of the response by adding solutions only if they present a new and different perspective from their predecessors.

The proposed approach (combining SimPO with FCS+Reflection) succeeded in significantly reducing the number of generated tokens (e.g., a 48.6% reduction on MATH500) while maintaining and even slightly improving accuracy across various benchmarks with different difficulty levels (GSM8K, MATH500, GPQA, AIME).

Explanation of Concepts:

SimPO (Simple Preference Optimization): This is the training method used for fine-tuning the model. Its goal is to teach the model to prefer certain types of responses (in this case, more efficient ones) over others (less efficient). The paper found SimPO to be the most effective among the tested preference optimization methods.

FCS+Reflection (First-Correct Solutions + Reflection): This is the strategy used to create the training dataset for SimPO. In this method, the original model responses were taken and "simplified" by keeping only the minimal part of the response that led to the first correct answer (FCS), plus the second solution that arrived at the same correct answer (the Reflection part). The goal was to create "good" training examples that are both efficient (not too long) and retain the model's "long-thinking" or reflective capability.

Link to paper: https://arxiv.org/abs/2412.21187

Mike's Daily Paper: 06.05.25
Graph Generative Pre-trained Transformer

We're used to seeing language models trained in an unsupervised fashion (typically referred to as pretraining) on text. This paper extends the concept of generative model pretraining to graphs. Essentially, it converts a graph into a kind of text—that is, a one-dimensional sequence of tokens—and trains a transformer on that sequence. However, unlike text, a graph is inherently non-linear, and representing it as a sequence is not entirely trivial.

The approach chosen by the authors seems fairly intuitive: the graph is represented as a sequence of nodes and edges. Each node is represented as a pair consisting of its category (discrete) and its index. Each edge is represented as a triplet that includes the two nodes it connects and the edge type. The order of nodes can be arbitrary (i.e., invariant to permutation), but the node order is selected using a simple algorithm: first, pick the node with the lowest degree, and from its edges, choose the one that leads to the node with the smallest degree among its neighbors. Then remove this edge and repeat the process until all edges have been removed.

Once the graph is written as a sequence of nodes and edges (with a special token separating them), positional encoding (PE) is applied. The paper uses absolute positional encoding, where each node and edge is encoded based on its position in the sequence (the paper doesn't elaborate on the exact form of PE used). Then, training proceeds similarly to a language model—that is, autoregressive prediction: predicting the next token (node or edge) given the past tokens. In short, standard generative model training.

After pretraining, the paper proposes a rejection sampling-based fine-tuning approach. Suppose we want to generate graphs that meet a specific condition, and we had some such graphs during pretraining. We begin generating graphs and collect a dataset of those that satisfy the condition. Once enough are gathered, we fine-tune the model. This process—generation, filtering, and fine-tuning—is repeated.

The authors also propose a method for training with Proximal Policy Optimization (PPO) on graphs for a given reward function. The paper suggests combining the PPO loss, the critic loss (an estimate of the value function), and the original pretraining loss described earlier.

The paper is quite interesting, but one thing that concerns me about this approach is the invariance of this representation to permutations of the nodes. I believe this requires extremely intensive training over a massive number of node permutations, especially for large graphs. Otherwise, the node representations will be sensitive to order and not particularly effective.

https://arxiv.org/abs/2501.01073

Mike's Daily Paper: 08.05.25
Memory Layers at Scale

This paper caught my eye because it uses the word *"memory"* in the context of language models. Even today, when you interact with ChatGPT, Claude, or other models, you're not just speaking to a standalone language model—you're interacting with a full system that already includes memory layers (often implemented as Retrieval-Augmented Generation or caches). This paper proposes a neural network layer that functions as a memory mechanism—allowing both storage and retrieval based on a query.

This memory layer is conceptually similar to a transformer block, but with a key difference: instead of attending to all vectors in memory, it retrieves only those relevant to a query vector q. In other words, rather than mixing over the entire memory, the model selects the top k vectors most similar to q. This mechanism resembles a Mixture of Experts (MoE), where only a subset of feedforward (FFN) submatrices are activated. The difference is that in MoE, the "experts" are predefined FFN submatrices, while in the proposed method, *any* combination of FFN columns can be selected dynamically.

Given a query vector q, the model selects the top k most similar vectors from memory. These are then combined with a value matrix V (the paper doesn't detail exactly how—so I suspect it's a standard dot product). The result is then multiplied by a learned matrix W_1 after applying the SiLU activation (which has become quite popular recently). That output is finally multiplied by another learned matrix W_2.

Because we want to store a large number of vectors in memory, computing all similarities with q can be computationally expensive. As is common nowadays, the authors *shard the memory*

across multiple GPUs, compute similarities in parallel, and then merge the top results globally to select the most relevant memory vectors. Naturally, each GPU holds a smaller memory matrix, and the query vector q is also split into sub-vectors across GPUs.

This layer can be combined with transformer blocks in language models, but I see no reason it couldn't also be integrated with other architectures like Mamba. A light, enjoyable read—and along the way, I discovered a neat trick for efficient retrieval from distributed GPU memory.

https://arxiv.org/abs/2412.09764

Mike's Daily Paper: 11.05.25
EfficientQAT: Efficient Quantization-Aware Training for Large Language Models

Quantization-Aware Training (QAT) is a technique where the model learns during training to cope with the quantization constraints that will be applied at inference time. These constraints typically involve lower-precision computations (e.g., INT8 instead of FP32). With QAT, the model is trained by simulating the quantization process throughout, so that at every training step, computations emulate reduced-precision data processing. While accurate representations are retained for gradient computations, a form of controlled noise is introduced via quantization and dequantization operations to simulate the model's post-quantization behavior. This gradually pushes the weights and activations to become robust to quantization errors.

Unlike post-training quantization (PTQ), which does not adjust the model's parameters, QAT maintains performance much closer to the original full-precision model, even after converting to a quantized representation. Typically, fake quantization is used to simulate quantized operations within the computation graph, while retaining high-resolution gradients for backpropagation. This approach enables deployment of models on resource-constrained hardware like mobile and edge devices, without compromising prediction quality.

Quantization and dequantization in QAT rely on two key trainable parameters: the zero-point *z* (of the quantized representation) and a scaling factor *s* for a given target bit-width (say, 8-bit). The paper proposes two main innovations. First, training separate *s* and *z* parameters for each Transformer block (alongside its weights). That is, training starts with the first block and its corresponding *s* and *z*, which are then frozen before moving to the next block. Optionally, different *s* and *z* values can be trained for different layers within the block (e.g., attention and FFN).

The second innovation is full-model fine-tuning after the per-block phase. In this stage, *z* is kept fixed, and only the scaling factors s are updated.

That's it — and yes, the paper reports performance improvements of course…
https://arxiv.org/abs/2407.11062

Certainly — here is a stylistically refined version of your translated article, preserving the original meaning:

Mike's Daily Paper – 16.05.25
ICLR: In-Context Learning of Representations

Language models are capable of much more than merely recalling facts or following instructions — they can adapt their internal representations based solely on context, without any change to their weights (in-context learning). The paper we're looking at today demonstrates that language models can fully reorganize their internal semantic geometry using only a prompt, with weights held constant. And this isn't some "surface-level" shift in output — it's a deep restructuring of the model's internal representational space, driven entirely by the structure of the context.

Pretraining builds stable semantic spaces in language models: synonyms cluster together, countries form geopolitical groupings, and days of the week appear arranged in a circle. But what happens when a prompt disrupts those relationships? Can the model generate entirely new meaning from context alone? That's the core question explored in this paper. The authors deliberately strip away any pretrained semantic clues and compel the model to infer meaning solely from the structure of token sequences in the prompt — revealing a surprising ability to learn a new representational geometry during inference.

The experimental setup is as follows: they construct a graph where each node is a known token (a word like apple or train). A random walk on this graph produces token sequences that serve as prompts. The model is tasked with predicting the next node (token), even though the words themselves carry no direct semantic hints. For example, orange may be adjacent to train in the graph, yet far away (in terms of minimal path length) from pear.

To succeed in this task, the model must infer the graph structure and reorganize its token representations accordingly. The structure is encoded in the sampled token sequences — not in the individual words. When examining the internal activations across transformer layers, the

researchers observe a fascinating dynamic: at first (with short sequences), the token representations still reflect the pretrained meanings. But as context length increases (longer sequences), a sudden transformation occurs: tokens that are nearby in the graph begin to converge in the representational space.

This is not a gradual shift but a sharp transition — a kind of phase change. Below a critical context length (i.e., a certain number of sampled sequences from the word graph), the original meanings dominate. Once that threshold is crossed, the model “commits” to the new structure and remaps its internal world according to the hidden structure in the prompt. This demonstrates that the model is not just memorizing token pairs. It isn’t merely retrieving the next word from memory — it’s constructing a consistent, comprehensive structure (starting from fairly early transformer layers) based on local patterns. One of the key arguments of the paper is that shallow changes relying on memorization alone cannot match the model’s performance or reproduce the emergent geometry.

Let’s unpack that further. Imagine building a graph where nodes represent words, and edge weights are distances between their representations in the model (say, in a specific transformer layer). It turns out this graph is spectrally isomorphic to the graph used to sample words for the prompt. In other words, if we compute the principal component (PC — essentially the dominant eigenvector) of the spectral representations (roughly a weighted adjacency matrix) for both graphs, they produce similar results.

Specifically, if we construct graphs where the distance between any two nodes (words) is defined by the distance between their values along the first PC dimension, the resulting structures are strikingly similar. That is, the adjacency matrices of the two graphs — one representing distances between the model's learned word representations, the other being the original proximity graph from which prompts are sampled — are closely aligned. This is remarkable. It shows that the model’s internal representations are genuinely learning the core structure of the proximity graph embedded in the prompt.

The authors found another intriguing phenomenon: when using words with strong pretrained semantic meaning (like days of the week), the model doesn’t erase that prior knowledge. Instead, it retains the original proximity structure in the leading principal components of the representational space, while embedding the new prompt-induced structure in subsequent (but still meaningful) dimensions. This enables the model to simultaneously hold both the original and the new meanings — by geometrically separating them into different subspaces.

The authors compare this transition to percolation in physics: local connections accumulate until a critical threshold is crossed, at which point a global pattern abruptly emerges. Here, it's the length of context, not the size of the model, that dictates the emergence of the new structure. As the prompt length increases, the internal structure becomes increasingly predictable — until a sudden jump occurs. This research shifts our understanding of in-context learning. It shows that models don't just alter their outputs in response to context — they reconstruct their inner world to match the structure implied by it.

https://arxiv.org/abs/2501.00070

Mike's Daily Paper: 2025-05-17
ZEROSEARCH: Incentivize the Search Capability of LLMs without Searching

The ZEROSEARCH paper introduces a new method for training the search capabilities of large language models using reinforcement learning (RL), without relying on actual search engines. Instead of querying Google or any external API, they train a smaller language model to act as a *simulated* search engine, providing either relevant or noisy documents as needed.

The simulated model is fine-tuned(SFT) on query-answer pairs taken from real interactions with a search engine. Documents that led to correct answers are marked as positive, and others as negative (BTW, this is quite original, since such training on positive-negative pairs is usually done using RL rather than supervised fine-tuning). To support this training(SFT), the original question and correct answer are embedded into the prompt, allowing the model to learn deeper semantic connections and better control the quality of the returned results.

During RL training, the main model generates search queries, receives documents from the simulated model, and performs reasoning to produce an answer.

To enhance reasoning skills, they introduce a *curriculum learning* mechanism: the quality of documents gradually decreases over training, forcing the model to handle partial or incorrect information. Training is done using algorithms like PPO and GRPO, which are fairly standard.

The result: a system that matches—or even surpasses—the performance of models that rely on Google, with no API costs and full control over the information quality.

A nice paper on how to search without searching using LLMs.

https://arxiv.org/abs/2505.04588

Mike's Daily Paper – 24.05.25
rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking

It's been a few days since my last paper review, but I couldn't skip one on my birthday — even with the insane workload. Today, I'm covering a pretty interesting paper that came out four months ago. It combines fine-tuning a language model for mathematical tasks using **MCTS** which is short for **Monte Carlo Tree Search**. Most of you probably know MCTS from the famous DeepMind projects *AlphaGo* and *AlphaZero*, which trained models to play the game Go. Worth noting: AlphaZero learned to play entirely by playing against itself, without any prior knowledge of Go. The models developed were so powerful that one of them led the world champion in Go to retire (can't recall which version exactly). The mathematical core behind these models? MCTS.

So, what is MCTS? It's a search algorithm primarily used in games(before 2024) to make optimal decisions. It builds a decision tree by running numerous random simulations from the current state and evaluates the quality of possible moves. It then picks the move with the best average outcome over those simulations. The algorithm cleverly balances between exploring new moves (which may turn out to be efficient) and exploiting moves already proven effective in past simulations i.e., moves that usually lead to a win.

This process is repeated iteratively: in each iteration, the tree is expanded, and the quality scores of the moves are updated until a final decision is made. The four main steps in each iteration are: **selection** (choosing the next node), **expansion** (growing the tree), **simulation** (playing out the game), and **backpropagation** (updating node values back to the root). MCTS succeeds because it efficiently focuses on the most promising regions of the search tree, even when the search space is massive. In the end, given a game trajectory (a path in the tree), the model selects the node with the highest win probability.

But how does this relate to language models? The answer: **autoregressiveness**. Language models, like Go players, generate token-by-token namely step-by-step. The big idea in applying MCTS to language models is to use tree search to generate **high-quality datasets**. But unlike game trees where nodes are game moves, here each node represents a **step in a reasoning process**. This reasoning trace is then used to fine-tune (SFT) the model. So the question becomes: how do we sample correct and diverse solutions using this approach?

As the paper suggests, we use MCTS to train a model to solve math problems where we have a **clear reward signal** (is the solution correct or not) at the end. The challenge is that the **reward for intermediate steps is not obvious**. (By the way, PPO suffers from a similar issue: we train a reward model, but it only scores entire solutions, so we train a value function to estimate intermediate rewards which is essentially a regression problem.)

In MCTS, it's critical to construct a function that scores partial solutions (nodes), otherwise, the tree search won't work well. Every node in the tree is generated by sampling from the language model. At first, a node's score is based on how frequently it appears in correct solutions: the more often it shows up in reasoning chains that lead to a correct answer, the higher its score. Later (once this scoring function stabilizes), the paper adopts an approach similar to training a reward model in RLHF: at each depth level of the tree, they select the highest- and lowest-scoring nodes and train a node-ranking model using a **Bradley-Terry** method (as is standard in RLHF). This score function is then used in a **UCT**-based selection strategy (Upper Confidence bounds for Trees), which balances exploration and exploitation

.

To boost quality and enforce precision, the model is required to **execute each reasoning step as Python code**, and if the code fails the test, the node is discarded.

The process starts with a small language model, builds a solution tree (with all the steps described), selects the best solutions (highest-scoring ones), runs SFT on the model, and repeats. The end result? A small, lightweight model that can solve fairly complex math problems (at least according to the paper claims).

https://arxiv.org/abs/2501.04519

Mike's Daily Paper – 26.05.25
Neuro-Symbolic AI in 2024: A Systematic Review

This survey paper reviews the rapid evolution of Neuro-Symbolic AI (NSAI) over the last half-decade. Drawing from a methodically filtered corpus of 158 peer-reviewed works, the authors provide a taxonomically grounded, thematically structured map of where the research community has concentrated its efforts—and where it has conspicuously not.

A decisive 63% of the literature reviewed pertains to learning and inference, underscoring the field's present commitment to blending statistical learning with logic-based constraints. Techniques such as Logical Neural Networks, symbolic priors in few-shot learning, and loss functions infused with semantic penalties exemplify attempts to ground neural learning in symbolic structure. These approaches not only improve sample efficiency and generalization but begin to bridge the conceptual gap between inductive pattern extraction and deductive reasoning.

Knowledge representation emerges as another strong pillar, accounting for 44% of the literature. The best work in this area transcends simple graph embeddings, aiming instead to encode dynamic, structured, and commonsense knowledge into systems that can adapt over time. Systems like NeuroQL point to an interest in building models that are not just knowledgeable, but epistemologically structured—models that can “do more with less” while maintaining semantic alignment.

Closely interlinked is logic and reasoning (35%), where models such as DeepStochLog and Logical Credal Networks attempt to recast logic programming in neural terms. These contributions enable hybrid reasoning over uncertainty, a core capacity if Neuro-Symbolic systems are to function in complex, partially observable environments. The coupling of probabilistic logic with symbolic control architectures is a recurring and fruitful pattern.

Despite growing deployment of AI systems in critical applications, only 28% of the reviewed works tackle explainability and trustworthiness. This imbalance is not merely statistical but structural. Most advances focus on post hoc summarization, semantic-level revision, or improved document-level inference. Notable contributions include Braid, a reasoner that blends probabilistic and symbolic rules to reduce brittleness, and FactPEGASUS, which aligns LLM-generated summaries with factuality through hybrid training paradigms.

However, the broader vision—building systems whose reasoning is not just intelligible after the fact but inherently transparent—remains elusive. There is little systematic exploration of how explainability can be baked into learning objectives, model architecture, or inference procedures. The result is a disconnect: systems grow increasingly powerful, yet their inner workings remain opaque to their creators.

Perhaps the most significant contribution of the paper is its elevation of Meta-Cognition from a footnote in prior taxonomies to a defined and urgent subdomain. Only 5% of the papers reviewed touch this theme, yet the concept of self-monitoring, adaptive control, and introspective reasoning is arguably the missing piece in current Neuro-Symbolic architectures.

The paper adopts a rigorous definition of Meta-Cognition as the capacity to regulate and reflect on internal cognitive processes. In practical terms, this encompasses symbolic controllers layered atop RL agents, integrations of LLMs with cognitive architectures (e.g., ACT-R, Soar), and theoretical frameworks aligned with the Common Model of Cognition. These works, though few, gesture toward a future in which AI systems are not only reactive but strategically

self-aware and capable of managing attention, choosing among inference strategies, and self-correcting in unfamiliar contexts.

Importantly, the authors push beyond Kahneman's dual-system metaphor, noting that human cognition is neither binary nor layered in isolation but a densely interconnected "system of systems." To build Neuro-Symbolic agents that mirror human adaptability, Meta-Cognition must not be an afterthought—it must be the glue binding perception, reasoning, learning, and communication.

One of the most revealing observations is the rarity of work that integrates all four core domains: learning and inference, logic and reasoning, knowledge representation, and explainability. The only flagship example is AlphaGeometry, a system that synthesizes millions of geometry problems using a neural generator and then guides a symbolic solver using large-scale pretraining. The system represents a blueprint: domain-specific expertise, procedural abstraction, symbolic tractability, and model guidance through learned heuristics. However, it remains an outlier.

The limited overlap between explainability and other domains is especially concerning. Integration remains low, suggesting that while individual capabilities are advancing, their coordination into coherent, trustworthy systems is still in its infancy. If Neuro-Symbolic AI is to realize its promise as a human-aligned intelligence paradigm, this fragmentation must be addressed.

Methodologically, the paper's filtering approach is a strength in itself. Of 1,428 initial papers, only 158 made the final cut—selected for relevance, peer review status, and the availability of open-source code. This stringent filtering underscores a deep concern with reproducibility and practical utility, ensuring that the findings reflect not just speculative ideas but tested, shareable systems.

By including only papers with publicly available code (with exception for Meta-Cognition due to scarcity), the authors offer a baseline for transparency and reuse. This also implicitly critiques the still-too-common practice of publishing complex Neuro-Symbolic proposals without operational instantiations.

This review represents a major milestone in mapping the Neuro-Symbolic AI landscape. It identifies what is working: structured learning with symbolic priors, logic-based generalization, and hybrid representation frameworks. It also calls out what is missing: native explainability, introspective control, and integrated systems capable of autonomous adaptation.

Above all, it posits Meta-Cognition not as an aspirational ideal but as a foundational requirement. If Neuro-Symbolic AI aims to move beyond brittle heuristics and monolithic networks, it must grapple with the architecture of thought itself—not just its outputs. This is a field on the cusp. The research community has assembled the pieces. The question now is whether it can build the system.

https://arxiv.org/abs/2501.05435

Mike's Daily Post: 28.05.25
Jasper and Stella: Distillation of SOTA Embedding Models

A pretty interesting paper proposing a relatively simple, but apparently effective, method for distilling knowledge from several large multimodal teacher models into a single small model (student). The rationale, of course, is that the small model will (hopefully) manage to learn the rich representations from two large models on the one hand, while remaining small on the other, which is great, since it makes it easier to use with RAG systems. After all, models with lower-dimensional representations require fewer arithmetic operations to compute similarity between a given data representation and those stored in the RAG.

The distillation process takes place in 3 main stages:

Stage 1:

The authors attempt to align the representations produced by concatenating (and normalizing) the outputs of

several teacher models (the big and strong ones) with those of a single small model. In this stage, the representation vector dimension produced by the student model is set equal to the sum of the dimensions of the teacher models' representations.

The loss function has three components:

1. Dot Product Alignment: It tries to make the dot product between the concatenated teacher representation and the student representation equal to 1 (i.e., they're aligned).

2. Correlation Alignment: It aligns the correlations between representations of different data samples by minimizing the squared distance between the “uncentered covariance matrices” of the teachers and the student. Here, the uncentered covariance is the product of a matrix containing the batch's data representations and its transpose.

3. Contrastive Loss: This encourages the student model to bring similar samples (as defined by the teacher) closer in embedding space, while pushing apart representations of dissimilar data.

Stage 2:

They reduce the embedding dimension of the student model (which was initially equal to the sum of the teacher models' embedding dimensions) while preserving its representational quality. How? By adding three new layers to the student model from the previous stage, and training only these new layers using the last two losses from Stage 1.

Stage 3:

They train the visual encoder (image side) of the multimodal student model, while freezing all other parts. The goal here is to align the representation of an image, produced by the visual encoder, with the representation of the image caption, produced by the teacher models. They reuse the three losses from Stage 1 in this step.

That’s it — a light and easy-to-digest paper...
https://arxiv.org/abs/2412.19048

Mike’s Daily Paper – 30.05.25
Learn Beyond the Answer: Training Language Models with Reflection for Mathematical Reasoning

Exactly one year ago (30.05.24), I started the daily paper review and launched a Telegram channel(Science and AI with Mike) for it. Since then, I’ve written 253 reviews—that’s roughly one every 1.44 days, although I’ve slowed down a bit recently.

So to mark the date, I chose to review a paper that's just under a year old on an important topic…

This paper presents a significant conceptual and methodological innovation (as of 10 months ago) in training models for mathematical reasoning. Unlike most prior work, which focused on

enriching datasets through additional questions or alternate answers (i.e., augmentations along the example dimension), the core innovation here is the introduction of an entirely new approach: Reflective Augmentation.

Instead of adding more examples, the authors propose deepening each existing example by appending a reflective section after the standard solution. This reflection includes two main components:

- Alternative Approach: A different solution offering a new perspective on the same problem.
- Follow-up – A generalized or analogous question designed to deepen understanding.

The novelty here is not merely technical but is cognitive. Rather than simulating learning through quantity (more questions), this method mimics qualitative learning, akin to how humans study, namely revisiting and reflecting on a given solution. A major strength of this approach is that it does not alter the inference-time process. That is, there's no need to generate or read the reflection section at runtime, yet it meaningfully shapes the reasoning learned during training. This marks a profound shift from "learning answers" to "learning principles through reflection."

The paper demonstrates empirically that this method not only improves accuracy on standard math problems but, crucially, yields exceptional gains in reflective scenarios: error correction, follow-up problem solving, and leveraging external feedback. Moreover, the approach is complementary to existing data augmentation techniques (such as Q-aug or A-aug), and combining them with Reflective Augmentation produced the highest results in the experiments.

In summary, the paper's innovation lies in three dimensions:

- A shift from solution memorization to principle-based reasoning through reflection.
- Emulation of human-like learning in training language models.
- A practical technique that can be applied to existing datasets without changing the inference pipeline.

This innovation has had wide-reaching influence, as seen in dozens of papers published over the past year that built on it but particularly in the domain of LLM-based mathematical problem solving, but also in designing interactive agents requiring flexible, non-linear reasoning. https://arxiv.org/abs/2406.12050

Mike's Daily Paper: 01.06.25
Common Sense Is (Still) All You Need

Disclaimer: This is a review of an opinion paper and does not necessarily represent the reviewer's position.

In recent years, AI has made major advances—from conversational agents and image generation to self-driving cars. Yet one basic ability still separates machines from the kind of intuitive, adaptive intelligence shown by even the simplest animals: common sense. In *Common Sense Is All You Need*, Hugo Latapie argues that this missing capability is not just an inconvenience but is the core obstacle keeping AI from achieving real autonomy. He makes the case that if we want AI to safely operate in dynamic, unpredictable environments, then common sense must be designed in from the ground up. Bigger models, more data, and better benchmarks won't close the gap. Instead, we need a shift in how we build, train, and evaluate AI systems.

The paper's core argument is that today's AI efforts are skewed toward performance and scale at the expense of general understanding. Latapie challenges the field to move away from narrowly optimized models and benchmarks, and instead focus on what actually enables humans and animals to learn and act effectively in the real world. This means prioritizing flexible learning, adaptation, and contextual reasoning and not just better language completion or object recognition. It's a call to reorient the foundations of AI around the very qualities we take for granted in natural intelligence.

What's New and Important Here?

One of the paper's most original contributions is its expansion of the concept of embodiment in AI. Typically, "embodiment" refers to robots or agents with physical bodies that learn by interacting with the environment: picking up objects, walking, or navigating spaces. Latapie broadens this idea by introducing the notion of *cognitive embodiment*, where AI systems engage not only with physical environments but also with structured, abstract ones. In this view, interacting with a constrained puzzle or logic problem can be as cognitively rich as exploring a room. An AI that genuinely embodies an abstract environment can learn to reason through trial and feedback, rather than just recognize patterns.

A good example of this is the Abstraction and Reasoning Corpus (ARC), a benchmark composed of visual transformation puzzles. Each task presents a small set of input-output grid pairs, and the AI must figure out the rule and apply it to a new grid. While ARC is meant to test reasoning, many AI systems today "solve" it by memorizing patterns from large sets of training examples. Latapie argues that this misses the point: to really test for reasoning, AI should approach these problems with minimal prior knowledge and learn from the structure of the task itself. That's cognitive embodiment in action and it's central to his proposal.

Another key idea is the importance of starting from a blank slate or as Latapie calls it, *tabula rasa*. Most modern AI models depend on pretraining with massive datasets, which makes them

powerful in familiar situations but poor at generalizing to new ones. In contrast, humans and animals regularly encounter novel environments and must figure things out on the fly. Latapie argues that real autonomy requires this kind of flexible, low-assumption learning. AI systems should begin with very little preloaded information and build up knowledge as they interact with the world or task at hand. This would prevent overfitting to specific training sets and encourage deeper, more transferable reasoning.

Perhaps the most piercing critique in the paper is what Latapie describes as the “magic happens here” problem. In many current AI development workflows, the transition from advanced pattern recognition to general intelligence is assumed to somehow emerge from scale. If we just train bigger models, with more data and compute, autonomy will arrive… eventually. But Latapie calls this wishful thinking. Without common sense reasoning explicitly built into the architecture and training process, we’re merely pushing against the ceiling of diminishing returns. Progress appears smooth at first, but eventually hits a wall. The core issue, which is the absence of real-world understanding, still remains unsolved.

Benchmarks and What’s Wrong With Them

Latapie dedicates a substantial part of the paper to analyzing the limits of popular benchmarks and evaluation methods, using three high-profile examples. First is the ARC challenge, which was designed to test abstraction and reasoning rather than memorization. However, in practice, many AI models tackle ARC by relying on hundreds of training examples, and sometimes even the test problems leak into development workflows. This creates the illusion of general intelligence, when in fact models are just pattern-matching. Latapie proposes redesigning ARC to truly enforce minimal prior knowledge and test the system’s ability to reason in real time from a small set of examples.

Next, he looks at the case of full self-driving (FSD) systems, which are often benchmarked using the SAE autonomy levels, from Level 1 (basic driver assistance) to Level 5 (complete autonomy in all conditions). Today’s systems are largely capped at Levels 2 or 3, and Level 4 vehicles often rely on remote human assistance for unusual situations. According to Latapie, this ceiling is not just a hardware or sensor problem but ’s a reasoning problem. Without the ability to understand context, recognize unusual patterns, and make intuitive decisions, vehicles will always require fallback mechanisms. Common sense and not just better cameras or maps, is what separates current AI from the full autonomy of a human driver.

Finally, Latapie critiques the well-known Turing Test, which evaluates a machine’s ability to engage in a conversation indistinguishable from that of a human. While passing the Turing Test can be impressive, he argues that it doesn’t indicate true understanding. A chatbot might produce fluent answers using statistical associations, without any grasp of the meaning or context behind its words. Such systems may appear smart in controlled conversation but would fail catastrophically in environments that require reasoning, decision-making, or adaptability. In Latapie’s view, these benchmarks risk distracting the field from what really matters: developing grounded, reasoning-based intelligence.

Scaling Isn't the Answer (Alone)

Latapie acknowledges that scaling has brought genuine breakthroughs, particularly in language and vision. Larger models have unlocked impressive capabilities in text generation, translation, and image analysis. But he also highlights a growing problem: scaling is starting to hit diminishing returns. Several real-world benchmarks have stalled despite increases in compute and dataset size. For instance, in object recognition (COCO), anomaly detection in video (UCF-Crime), and action localization in video (ActivityNet), performance improvements have plateaued even as models grow more complex.

This suggests that we may be nearing the practical limits of what scaling alone can achieve. The implication is that future progress will depend less on size and more on solving deeper architectural challenges, like how to build systems that can learn, adapt, and reason across unfamiliar situations. Scaling gives us sharper tools, but without common sense, those tools remain narrowly useful and brittle when faced with real-world complexity.

Theoretical Rigor (Made Simple)

The paper also tackles several famous theoretical issues in AI, explaining how common sense could help address them. One is the No Free Lunch Theorem, which says that no learning algorithm is best for every possible problem. Latapie's response is practical: we don't need to solve *every* problem but just the ones that matter. By focusing on well-structured domains with clear patterns and rules, AI systems can learn efficiently without violating the theorem's limits.

He also addresses the Frame Problem, which refers to the difficulty of figuring out what matters and what doesn't in any given situation. Latapie suggests that embodied interaction, whether in the physical world or in structured tasks, helps AI systems learn relevance through experience. Similarly, the Qualification Problem, which points to the challenge of listing all the preconditions for an action, can be managed through adaptive reasoning and learning from trial and error. These responses ground abstract philosophical concerns in concrete design strategies, showing how they can be addressed through better AI architecture rather than brute-force data accumulation.

How Should We Build AI Instead?

To move forward, Latapie proposes a series of practical changes in how we build and evaluate AI. First, benchmarks need to be redesigned to test reasoning, not recognition. That means limiting the prior knowledge an AI system is allowed to use, forcing it to engage with each problem as new. Evaluation should shift away from just scoring correct answers and toward understanding *how* the system reached its conclusion, meaning that process matters more than output.

Architecturally, Latapie advocates for integrating symbolic logic with learning-based approaches. This hybrid strategy would combine the structure and transparency of logic with the flexibility of statistical models. He also emphasizes the need to draw from neuroscience and cognitive

science, namely to build architectures that reflect how biological systems acquire and apply common sense. In short, the AI software stack needs to evolve. Adding more layers to today's models won't be enough. We need to rethink what the base layers should even be.

What Happens If We Don't Change?

Latapie warns that continuing along the current path: scaling up models, polishing benchmarks, and assuming that autonomy will emerge, risks slowing progress and damaging trust. AI systems that look good in demos but fail in messy, real-world conditions can lead to disillusionment among users, developers, and investors. Worse, systems without common sense may behave in unpredictable or unsafe ways, especially as they gain more autonomy and the ability to self-modify. If these systems lack grounding in human values or context awareness, they could easily make decisions that appear intelligent but cause real harm.

The paper argues that public fears around AI, especially fears of superintelligent systems behaving recklessly, are really fears of intelligence *without* common sense. And those fears are not unfounded. Building AI that understands its environment, learns responsibly, and reasons about consequences is not just a technical goal but is a safety requirement.

https://arxiv.org/pdf/2501.06642

Yaniv's and Mike's Daily Paper: 03.06.2025
Reinforcement Learning for Reasoning in Small LLMs: What Works and What Doesn't

Large language models (LLMs) like OpenAI's GPT-o1 have revolutionized reasoning tasks—from solving complex math problems to writing elegant code—thanks to massive computing resources and vast datasets in Reinforcement Learning training. But can smaller, budget-friendly models achieve similar feats? A recent paper titled "Reinforcement Learning for Reasoning in Small LLMs: What Works and What Doesn't" by Quy-Anh Dang and Chris Ngo explores exactly this question.

Why This Matters

Big models can reason well but demand huge computational resources, making them costly and impractical for widespread use. Smaller models (around 1 to 2 billion parameters) are cheaper and easier to deploy, but currently lag behind in complex reasoning performance. Dang and Ngo's goal is ambitious yet practical: boost the reasoning performance of small models, with minimal resources and minimal cost.

The Approach: Group Relative Policy Optimization (GRPO)

The researchers chose a 1.5-billion-parameter model, DeepSeek-R1-Distill-Qwen-1.5B, and fine-tuned it using **GRPO**, the current go-to algorithm for reasoning fine-tuning previously proven effective in massive models, to a much smaller scale.

To keep costs low, the training was tightly constrained in terms of resources: it used only 4 NVIDIA A40 GPUs, was limited to a 24-hour time window, and relied on a modest dataset of just 7,000 carefully selected math questions.

Surprising Results

Despite these extreme constraints, the performance improvements were remarkable:
AMC23 benchmark accuracy jumped from **63% to 80%**. **AIME24 benchmark** scores reached **46.7%**, notably beating OpenAI's powerful **o1-preview** model (44.6%).

Perhaps most impressively, the entire training run cost only about **$42**, several orders of magnitude cheaper than typical state-of-the-art methods. Amazing how far reasoning has come in such a short time.

What Exactly Did They Do?

They conducted three insightful experiments:

- **Experiment 1**: Trained with challenging, high-quality math problems. The model rapidly improved, but then quickly degraded due to unstable optimization and language drift.

- **Experiment 2**: Mixed easier problems with difficult ones, achieving higher initial stability and impressive peak performance, though still eventually spiralling towards instability.

- **Experiment 3**: Used a cosine reward to encourage shorter answer length, achieving improved stability and impressive performance. This corresponds to findings in the "DR GRPO" paper, uploaded to arxiv just a few days earlier, diagnosing a bias for longer answers in GRPO.

Limitations and Open Questions

- **Why Their Model Works So Well:** One of the paper's most surprising findings is how a small model, trained briefly on limited data, achieves such strong reasoning performance. The authors offer little intuition or analysis to explain why this works so well. This leaves open questions about how robust these gains are particularly on tasks beyond the narrow math benchmarks tested.

- **Language Drift:** The multilingual base model eventually produces non-English outputs, causing training instability for all variants.

- **Domain specificity:** Their evaluation was limited strictly to mathematical reasoning trying to find out whether this approach can be applied to broader reasoning tasks like science or coding?

What's Next?

This study demonstrates that powerful reasoning doesn't have to be expensive. The surprising success of such a small model suggests future work could explore **The benefit of GRPO variants** like DR GRPO in resource constrained settings, as well as **broader domains:** Evaluating performance across diverse reasoning tasks to see if small models can become universally capable. **Hyperparameters tuning** also seems like a necessary next step, as higher KL divergence loss could improve training stability.

Bottom Line

Dang and Ngo's paper reveals a promising pathway toward affordable, reasoning-capable language models. While the "why" remains elusive, the practical implications are profound: democratizing powerful AI reasoning to smaller research labs and organizations everywhere.

Read the full paper here →

Explore their code, it holds significant promise for enabling reasoning within your specific domain with minimal investment.

Mike's Daily Paper Review: 05.06.25
Task Singular Vectors: Reducing Task Interference in Model Merging

Today's review is short and light.

The paper addresses the problem of merging multiple models, each fine-tuned from the same base model on different tasks, into a single model that performs well across all tasks. Each fine-tuned model has undergone its own specific weight changes during training (e.g., via LoRA, though not necessarily).

A common baseline approach is to average the weight deltas and add the result to the base model. However, the authors point out that this naive strategy often performs poorly even when the tasks are similar. To address this, they propose an intuitive method designed to reduce **interference** between the delta matrices of different tasks.

What's the approach?
First, they observe that the delta matrices are typically low-rank. They apply **Singular Value Decomposition (SVD)** to each delta matrix, yielding orthogonal matrices U_i and V_i, and a diagonal matrix D_i of singular values. Then, for each delta matrix, they retain a small number of top singular vectors just as one would in PCA.

In the second step, they aim to **decorrelate** the update directions across tasks. To do this, they concatenate the U_i's and V_i's from all tasks into large matrices and compute whitening (decorrelation) transformations for each. This involves standard linear algebra techniques, including the Moore-Penrose pseudoinverse.

Finally, rather than summing the deltas directly, they apply the whitening transforms and construct a **weighted** combination of the update matrices. The goal is to reduce interference while preserving task-specific structure.

The method is applied separately to each layer—though it's unclear whether this per-layer treatment is novel.

https://arxiv.org/abs/2412.00081

Mike's Daily Paper Review: 07.06.25
Rate-In: Information-Driven Adaptive Dropout Rates for Improved Inference-Time Uncertainty Estimation

Today I'm reviewing a special paper on several levels. The first level: one of the co-authors of this paper is none other than Yann LeCun, one of the founding fathers of deep learning. The second level involves the well-known (but not well-known enough) Israeli researcher Ravid Shwartz-Ziv, who is also a professor at New York University. The third level is the paper's topic: uncertainty estimation for neural network predictions which is a subject that greatly interests me, yet I haven't reviewed a paper on it in some time.

How can we estimate the uncertainty in a neural network's predictions? There are several families of methods mentioned in the paper:

- **Bayesian Neural Networks (BNNs):** These define probabilistic distributions over the network weights, enabling uncertainty modeling through the posterior distribution. However, they are computationally heavy and hard to scale.

- **Ensemble Methods:** Multiple models are trained and their predictions are aggregated. These methods can capture both epistemic and aleatoric uncertainty but require considerable computational resources.

- **Test-Time Data Augmentation:** Applies perturbations to the input (e.g., rotation, blurring) during inference to approximate a predictive distribution. This is particularly

effective when there's prior knowledge of the data structure.

- **Noise Injection into the Model:** Controlled noise (e.g., Gaussian) is added to weights or activations to test sensitivity beyond simple input perturbations.

- **Monte Carlo (MC) Methods:** These use random sampling to estimate uncertainty. For example, MC Dropout applies dropout during inference as well, to sample the space of network weights. The paper mentions many additional MC-based techniques for uncertainty estimation in networks.

But how can one estimate the certainty itself? One approach is to use information-theoretic techniques to analyze how information flows within the network and how much that flow is “disrupted” by the methods above (e.g., MC Dropout). Broadly speaking: The more information flow is disrupted, the higher the prediction uncertainty.

Information-theoretic tools are quite common in deep learning research. For example:

- **The Information Bottleneck Principle (by Naftali Tishby):** Suggests that the layers in a neural network aim to compress the input while preserving relevant information for the output. It's used to analyze learning dynamics and generalization.

- **Mutual Information Analysis:** Estimating the mutual information between input, hidden layers, and output helps us understand how information flows and transforms in the network. This is the **specific technique used in this paper**.

- **Information-Theoretic Regularization Techniques:** Methods like **information dropout** regulate information flow during training to improve robustness and generalization.

Okay, so what does the paper propose? The paper introduces an MI-based enhancement of MC Dropout. Instead of using a fixed dropout rate across all layers (i.e., the proportion of neurons dropped in a layer), the authors propose to set the dropout rate adaptively, based on how much it disrupts information flow in each layer. The goal is to keep information loss per layer approximately constant. If the mutual information loss in a layer is too high (above some small threshold ϵ), they decrease the dropout rate. If the loss is too low, they increase it to inject more uncertainty.

By the way, this disruption to information flow is computed via the mutual information between the input activations to the layer and its output activations. This turns out to be non-trivial, and the paper discusses at length how to compute this effectively.

https://arxiv.org/abs/2412.07169

Yaniv's and Mike's Daily Paper: 09.06.25
Spurious Rewards: Rethinking Training Signals in RLVR – Fast Overview

One-line takeaway

Even completely noisy or adversarial rewards can unlock big math-reasoning gains but only for models whose pre-training already hides the right skills.

Why this matters

RL with Verifiable Rewards (RLVR) is the state-of-the-art way to teach LLMs step-by-step reasoning. The paper asks a blunt question: **how much of that "reward" signal do we actually need?** Spoiler: sometimes almost none.

Experimental recipe

The authors fine-tuned the Gwen-2.5-Math-7B model on a math dataset (DeepSeek-Math) using GRPO for 150 reinforcement learning steps. They tested five types of reward signals:

- **Ground-truth reward**: A standard signal where the model gets +1 only if the final answer is correct.
- **Majority-vote reward**: The model generates 64 responses, and the most common one is treated as the correct answer for reward purposes.
- **Format reward**: The model is rewarded just for including a boxed math expression (like `\boxed{}`), regardless of whether the answer is actually right.
- **Random reward**: The reward is assigned randomly with a 50% chance for +1 or 0, completely independent of the answer.
- **Incorrect reward**: The model is rewarded only when it gives an incorrect answer.

Surprisingly, all of these reward types led to major performance gains — even the random and incorrect rewards brought the model close to the performance of ground-truth training.

What they found

1. **Qwen models soar regardless of reward**: every weak/spurious signal above gives 16–26 pp gains, nearly matching gold supervision. Llama 3 and OLMo 2 barely move unless given the real label .
2. **Hidden skill = code reasoning:** Qwen-2.5-Math already solves 65 % of problems by writing inline Python (without execution). RLVR pushes that to ≈ 90 %, and accuracy rises in lock-step.

3. **Lang → Code switch explains the lift.** 58 % of Qwen's gain comes from questions that flip from pure language to code reasoning during RLVR .
4. **Random rewards work via optimizer bias.** GRPO's clipping term preferentially up-weights high-probability tokens, amplifying whatever strategy the model already prefers; ablate clipping and the random trick stops working .
5. **No free lunch across models.** Llama and OLMo lack useful code priors, so the same spurious signals either help little or hurt .

Take-home message

RLVR often **surfaces**, rather than teaches, reasoning that pre-training has already seeded.
This is in line with another recent paper showing RL reasoning results can be achieved with *a single* training sample. For reward design and future benchmarks, verify results on multiple model families and not just Qwen.

https://github.com/ruixin31/Rethink_RLVR/blob/main/paper/rethink-rlvr.pdf

Mike's Daily Paper: 2025-06-11
TRANSFORMER-SQUARED: SELF-ADAPTIVE LLMS

I've been intending to review this paper for a while, but it got lost in my never-ending paper review pipeline (currently standing at 353 papers waiting to be either reviewed or discarded). The paper was written by scientists (hopefully 🙂) from Sakana AI, a company that made headlines for releasing an AI Data Scientist (which, to the best of my recollection, received quite positive reviews). The paper proposes a very simple improvement to the training process of language models in multitasking scenarios.

Here, "multitasking" refers to training several expert models (not to be confused with MoEs – Mixture of Experts), each specializing in a specific task, derived from a strong base model. Each of these models is trained in a way similar to adapters, which are a type of **PEFT**, *Parameter-Efficient Fine Tuning*, meaning that only a small number of weights are updated during fine-tuning.

The paper introduces a PEFT method called **SVF**, which stands for *Singular Value Fine-Tuning*, aimed at adapting the model to a given task. As the name suggests, SVF is based on singular values and in this case, the singular values of the weight matrices in the MLP layers. Incidentally, each MLP in a transformer block contains two weight matrices, and the paper doesn't explain (or at least I didn't see it explained) exactly how the weight matrix is constructed in each block (perhaps they apply SVF to each matrix separately).

So what does SVF actually do? It performs an **SVD**, *Singular Value Decomposition*, on the weight matrices in each transformer block. One of these matrices is diagonal, while the other two are orthogonal (on the left and right). The authors insert a **learned diagonal matrix Z** into this product and train it during fine-tuning. There's an implicit assumption here that the base model has already "learned all possible tasks," and during fine-tuning, we only need to amplify those relevant to the current task.

Interestingly, the fine-tuning is done using a **reinforcement learning method** called **REINFORCE**, along with standard regularization commonly used in RL training of language models. That's right – they didn't use PPO, GRPO, or DPO. In addition, the authors applied this method to tasks with **verifiable rewards** – meaning tasks where you can definitively check if the answer is correct, such as math problems or coding. During such training, only the **Z matrices** are trained across all layers.

During **inference**, the authors propose three methods:

1. **Prompt-based task selection**: Ask an LLM (using a proper prompt) to determine which task a question belongs to. Based on the answer, run the model with the appropriate Z vector for the selected task.

2. **Discriminative classifier**: Train a discriminative model that identifies the task type for a given question.

3. **Linear task combination**: Assume a small dataset per task and train a Z-weight vector for each model (per task). Instead of assigning a question to a single task, represent it as a **linear combination of all tasks**. In the end, the task gets its own representation (via its own Z vectors).

https://arxiv.org/abs/2501.06252

Mike's Daily Article: 13.06.25
Inference-Time Scaling for Diffusion Models beyond Scaling Denoising Steps

The authors of this paper are doing something quite unconventional in the image domain. They ask a seemingly simple question: Suppose we've already trained an excellent diffusion model. Can we get more out of it at inference time? Is it possible to improve the quality of the generated image *without* just adding more and more denoising steps? Surprisingly, their answer is yes. But the way they get there is through a completely original idea: searching for *better noise*.

Anyone who has worked with diffusion models knows that the entire process unfolds from an initial noise vector. This vector is usually chosen randomly, and all it needs to be is “white noise.” But what if *not all noise is created equal*? What if we could choose *smarter* noise, noise that leads the model to generate higher-quality images, *without* changing the architecture, the number of steps, or the model’s weights?

This paper proposes exactly that: Instead of extending the diffusion trajectory (i.e., increasing the number of steps), we can invest the same computational budget into a selective search for initial noise that yields a better result. This represents a significant conceptual shift: we stop thinking of denoising as the only axis for quality improvement and begin to treat the stochasticity itself, i.e., the noise, as something not just to be sampled but to be *optimized*.

To make this work, two things are needed: First, a way to measure the quality of the final output. In this paper, that’s called a **verifier**, which can be a function like CLIPScore, an aesthetic predictor, FID, or any other quality metric aligned with the sampling objective. Second, a **search algorithm, namely** a method to generate or select new noise vectors, compare them based on the outputs they produce, and find one that gives a better result.

This structure, a verifier alongside a search method, is the core innovation here. It’s generic enough to be architecture-agnostic and doesn’t require fine-tuning. All you need is a scoring function and the ability to run a few samples. From there, you just start searching.

The search could be as simple as generating 64 noise vectors and picking the best one. Or it could be more sophisticated, like applying variations to a given noise vector in random directions (a method known as Zero-Order Optimization), or even modifying the diffusion path by adding noise at only certain stages and restarting from there, a technique they call Search-Over-Paths. In other words, this is not just about improving quality. It’s a new approach to understanding the *paths* diffusion models take, and how noise influences them.

But perhaps the most profound element of this paper is what it *doesn’t* try to do. It doesn’t suggest changing the model. It doesn’t claim the network needs to be better trained or re-architected. The entire innovation lies in recognizing that the choice of *which noise to start with* is an active, optimizable parameter at inference time. And that matters because until now, the inference stage of diffusion models has been treated as mostly static: efforts focused on

shortening it, or improving the denoising trajectory, but never on modifying the initial noise. This paper dismantles that assumption.

In a sense, what we're seeing here is the injection of algorithmic thinking into what was assumed to be a passive stage, meaning the point at which the model is already trained and we're "just running it." But once we accept that we can optimize inference-time parameters like the noise itself, we open the door not only to better outputs but also to a deeper understanding of diffusion's internal mechanics.

That's why I believe the main contribution of this paper isn't just another FID graph. It's in the *shift in mindset* it represents: from stochastic models treated as black boxes, to systems where the randomness itself becomes controllable, optimizable, and reconfigurable in real time. Will this become a common practice? Maybe only in high-end generative tasks where every small improvement counts. But as a concept, it's another step in transforming inference from a static process into an intelligent one and that's a fascinating move in its own right.

https://arxiv.org/abs/2501.09732

Mike's Daily Paper Review: 14.06.25
Is Stochastic Gradient Descent Effective? A PDE Perspective on Machine Learning Processes

This paper is quite heavy, but I tried to make the review accessible (I didn't dive *too* deep myself because the paper is genuinely complex).

There's something deceptively simple about Stochastic Gradient Descent, or SGD. For years, it's been the backbone of machine learning (ML), especially deep learning, yet the question of *why* it works has largely remained in the realm of vague intuition. Most explanations seem to circle back to fuzzy claims like "it finds flat minima" or "noise helps escape local minima." The paper I'm reviewing today tries to bring order to this, and unlike most work in the field, it offers a completely new angle: it describes SGD as a diffusion process evolving over time — seen through the lens of partial differential equations (PDEs).

The authors aim to shift our understanding of learning dynamics. No longer is it about tracking a single point in weight space rolling down a loss landscape. Instead, they propose describing the entire probability distribution of possible configurations namely the density over network weights space, as it evolves with time. If you come from mathematical physics, this will immediately remind you of the Fokker–Planck equation, which describes how particles diffuse in a system. The analogy here is powerful: the weights are like particles moving along the gradient of the loss function, with added noise from the stochastic nature of SGD (i.e., mini-batch selection).

What's striking is that this physical model doesn't just mimic what SGD does but it explains its success. For example, when analyzing the kinetic energy of the system, the stochastic "noise" from mini-batch selection turns out to be far more than a nuisance but plays a critical role in stability. It balances progress so that the system neither races too quickly nor gets stuck in unstable regions. The authors show that there are energy constraints dictating the pace of learning and that noise magnitude is directly tied to how deeply the system can descend the loss.

The paper also makes a sharp conceptual distinction between two views of learning:

- The local view that looks at parameter updates step by step, and
- The global view that describes the entire evolving distribution as a continuous probability flow in weight space.

Just like in physics, shifting from a pointwise to a distributed description uncovers insights that were previously hidden. Suddenly, we can ask not just *where* the weights are going, but *where* they're concentrating, *how* they spread, and *how* the structure of the loss function shapes that behavior.

One of the most impressive parts of the paper is the analysis of temporal recursion. The authors don't stop at showing that SGD converges but they investigate how its recursive structure, based on repeatedly following gradients, aligns with the continuous dynamics of the physical PDEs. This comparison between discrete-time recursion and continuous diffusion lets them articulate, for the first time, general principles about when SGD succeeds, when it might fail, and how we can control that behavior.

But what struck me the most is that this whole framing opens the door to future development. If we accept the paradigm that SGD is not merely a greedy downhill process, but a physical system governed by differential laws, then maybe we shouldn't focus on improving SGD per se. Instead, we could shift to PDE-guided training, where we directly model the desired evolution of the distribution and solve backwards to find the matching dynamics.

In that sense, this paper doesn't just explain SGD's past but offers a bold future for deep learning. A future where we're not blindly groping through high-dimensional surfaces, but designing dynamic systems with explicit physical structure. It's nothing short of a paradigm shift the one that may change how we approach optimization entirely.

https://arxiv.org/abs/2501.08425

Mike's Daily Article: 15.06.25
Random Teachers are Good Teachers

An old but very interesting paper at least in my opinion…

This paper presents an interesting and deeply counter-intuitive finding that challenges foundational assumptions in the fields of knowledge distillation and self-supervised learning (SSL). The authors demonstrate that a "student" model can learn high-quality representations by distilling knowledge from a "teacher" network whose weights are completely random and untrained. The work dismantles the standard "teacher-student relationship" to isolate and study the dynamics of learning with knowledge distillation, revealing a process similar to implicit regularization that is not dependent on the teacher possessing any actual "knowledge."

As mentioned, the main goal of the paper is to investigate the dynamics of knowledge distillation. The paper broadly examines two knowledge distillation regimes: with labeled data and without labeled data (label-free).

The core of the paper's contribution lies in its elegantly simple experimental setup. The authors create a scenario designed to remove various confounding factors that are typically credited with the success of distillation and SSL methods.

- **No "Dark Knowledge":** The teacher network is initialized randomly and is then "frozen." It is never exposed to the training data or labels, meaning it contains no learned information about the task or the data distribution. The student's goal is simply to minimize the KL-divergence between its output distribution and the teacher's static, random output (though sometimes the student's loss on the data is also added).
- **No Data Augmentation:** Unlike mainstream SSL methods, this work intentionally removes all data augmentations. This ensures that the learned invariances do not stem from the explicit inductive biases introduced by techniques like cropping, flipping, or color jittering.
- **No Labels:** The entire distillation process is fully unsupervised and label-free. The true class labels are only used at the very end to evaluate the quality of the learned representations via linear probing that is, training a linear classifier on top of the frozen representations from the student's encoder.

This minimalist framework ensures that any observed learning effect is attributable solely to the interaction between the model's architecture, the natural data distribution, and the gradient-based optimization dynamics of the teacher-student setup.

The results of the experiment are very surprising. The student network consistently and significantly outperforms its random teacher in terms of linear probing accuracy across many datasets (such as CIFAR-100, STL10, TinyImageNet) and various architectures (such as ResNet, VGG). In other words, the data is more important than the teacher.

Another finding in the paper is the "locality phenomenon": the initial proximity of the student's weights to the teacher's is critical for successful learning. The authors investigate this by initializing the student's weights as a convex combination of the teacher's weights and a set of random weights, controlled by a locality parameter α. When α is close to zero (meaning the student starts almost identically to the teacher), learning is fastest and the final performance is

highest (here the student has the same architecture as the teacher - this is not a practical scenario but is interesting for investigation).

This finding suggests an interesting geometry of the loss surface. The teacher's parameterization, θ_T, constitutes a trivial local minimum where the distillation loss is zero. However, the optimization process does not remain there. Instead, it finds a nearby, non-trivial local minimum, θ_S, which corresponds to a region with much higher accuracy (for the training dataset, meaning better representations). Visualizations of the landscape reveal that the teacher often sits within a sharp, "asymmetric valley." The student model appears to escape the trivial solution by moving towards the "flatter" side of this valley, a region whose geometry is known to correlate with better generalization.

Perhaps the most profound finding is that a student checkpoint, developed entirely without labels (only knowledge distillation), exhibits structural representations that were previously thought to appear only in the early stages of supervised training. That is, the student network approaches the teacher (even a random one) when it has "a winning ticket inside it" namely a small subnetwork that knows how to do the same thing.

- **Emergence of the Lottery Ticket Hypothesis:** The authors found that even at the first checkpoint, the student contains a "winning lottery ticket", a sparse subnetwork that can be retrained from its initial weights to achieve high accuracy on a supervised task. A randomly initialized network does not have this property; it appears in supervised networks only after several training epochs. This implies that distillation from a random teacher guides the network to a parameter configuration that is already structured for efficient learning.
- **Linear Mode Connectivity:** Typically, when an early student checkpoint is used as an initialization for several supervised training runs (each with different mini-batches), the resulting solutions are usually "linearly connected." This means one can linearly interpolate in the weight space between any two of these solutions without encountering a high loss barrier along the way. This stability indicates that the student has already converged to a "wide, flat basin" in the supervised loss landscape, effectively bypassing the chaotic initial phase of supervised optimization.

Conclusion

The paper presents a compelling case that the success of teacher-student frameworks is not solely attributable to the transfer of "dark knowledge" from a trained expert. Instead, the paper reveals that the implicit regularization created by the learning dynamics is a powerful engine for learning strong representations in its own right. By demonstrating that a network can develop sophisticated structural representations (like "winning lottery tickets") from a completely random signal, the authors force a re-evaluation of the fundamental mechanisms behind self-distillation and SSL. The work provides a testbed for future work aimed at demystifying the "early phase" of neural network training and the intricate geometry of their loss landscapes.

https://arxiv.org/abs/2302.12091

Mike's Daily Article: 16.06.25

Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap

In the rapidly evolving landscape of artificial intelligence, two paradigms – Large Language Models (LLMs) and Evolutionary Algorithms (EAs) – have often operated in parallel, each demonstrating formidable capabilities in their respective domains. LLMs have dazzled us with their generative prowess and understanding of natural language, while EAs have consistently proven their mettle in complex optimization and search problems, mimicking nature's ingenuity. But what happens when these two powerful forces begin to collaborate?

A recent paper, "Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap" (arXiv:2401.10034), plunges into this fascinating intersection, offering a comprehensive survey and a forward-looking roadmap for their synergistic integration. Far from being just another academic review, this work lays out a compelling vision for how LLMs and EAs can mutually enhance each other, unlocking new frontiers in AI development.

The central thesis of the paper is elegantly simple yet profoundly impactful: the relationship between LLMs and EAs is one of "reciprocal inspiration." This isn't just about throwing an LLM at an optimization problem or using an EA to tweak an LLM; it's about a deeper, more integrated synergy where each paradigm addresses the other's inherent limitations and amplifies its strengths.

The authors meticulously categorize this relationship into two primary directions:

1. **LLM-enhanced EA:** Here, Large Language Models are leveraged to improve various aspects of Evolutionary Algorithms. Imagine an LLM dynamically generating more diverse and relevant initial populations for an EA, or crafting sophisticated, context-aware fitness functions that are hard to hand-design. An LLM could act as a "problem understanding agent," interpreting complex problem descriptions to guide the EA's search, or even as a "repair mechanism," correcting invalid solutions generated by the EA. The potential is vast:

    - **Intelligent Initialization:** LLMs can generate diverse and promising starting points, guiding the EA away from purely random exploration.

    - **Adaptive Operators:** Designing crossover or mutation operators often requires domain expertise. LLMs could potentially generate or refine these operators on the fly based on the problem context.
    - **Fitness Function Engineering:** Crafting effective fitness functions is notoriously difficult. An LLM could assist by translating high-level objectives into quantifiable metrics or even generating code for evaluation.
    - **Explanation and Interpretability:** After an EA finds a solution, an LLM could generate human-readable explanations of *why* that solution is good or how it was derived.

2. **EA-enhanced LLM:** Conversely, Evolutionary Algorithms can bring their robust optimization capabilities to bear on Large Language Models. Training LLMs is computationally intensive and relies heavily on gradient descent, which can get stuck in local optima. EAs, known for their global search capabilities and ability to navigate non-differentiable spaces, offer an intriguing alternative or complement:
    - **Prompt Optimization:** EAs could evolve more effective prompts for LLMs, discovering nuanced phrasing that elicits superior responses for specific tasks. This goes beyond simple prompt engineering, enabling automated discovery of optimal prompts.
    - **Hyperparameter Tuning:** The myriad hyperparameters of LLMs (learning rates, batch sizes, architecture choices) could be optimized using EAs, potentially leading to more robust and efficient models.
    - **Neural Architecture Search (NAS):** EAs have a strong history in NAS. Applied to LLMs, they could discover novel, more efficient, or specialized architectures, particularly for smaller, more constrained models.
    - **Data Augmentation & Curation:** EAs could evolve strategies for selecting or generating training data that maximally benefits LLM performance, addressing data scarcity or quality issues.
    - **Robustness and Adversarial Attack Defenses:** EAs could be used to generate adversarial examples to test and improve LLM robustness, or to evolve defensive mechanisms.

The paper doesn't just theorize; it highlights "integrated synergy methods" across diverse scenarios, showcasing the practical implications of this collaboration. They touch upon:

- **Code Generation & Software Engineering:** Imagine an LLM generating initial code snippets, and an EA then optimizing that code for performance, efficiency, or even bug reduction. Conversely, an EA could suggest improvements to code structure, and an LLM could refactor the code based on those suggestions.
- **Neural Architecture Search (NAS):** This is a natural fit, as EAs have long been used to discover neural network architectures. Combining this with LLMs could mean an LLM proposes initial architectural motifs, which an EA then evolves and refines.

- **Various Generation Tasks:** Beyond code, think about creative writing, design, or even drug discovery. An LLM could generate initial ideas or structures, and an EA could then optimize these for specific criteria (e.g., novelty, coherence, effectiveness), leading to truly novel outputs.

This paper is incredibly timely and relevant. As LLMs become ubiquitous, understanding how to make them more robust, efficient, and intelligent, and how to harness them for complex problems, is paramount. The survey provides a crucial foundational step, systematically breaking down a complex emerging field. Its categorization of "reciprocal inspiration" offers a clear framework for future research and development. The authors' foresight in identifying challenges and proposing a roadmap is particularly valuable, guiding researchers toward the most promising avenues.

However, like any roadmap, its true value will lie in its execution. The challenges hinted at are significant:

- **Computational Cost:** Running EAs, especially for complex LLM tasks, can be prohibitively expensive. How do we make this integration efficient?
- **Representation Mismatch:** Bridging the gap between the discrete, symbolic nature of language (as handled by LLMs) and the continuous, numerical spaces often explored by EAs is non-trivial.
- **Interpretability of Synergy:** When an LLM and an EA collaborate, understanding *why* a particular solution was reached becomes even more opaque.
- **Defining "Optimal":** For many creative or complex problems, defining a precise fitness function for an EA, even with LLM assistance, remains a challenge.

What this paper truly highlights is the move beyond treating LLMs as isolated magic boxes. It pushes for a holistic view of AI, where different paradigm, each with its unique strengths, can be combined to overcome individual weaknesses. Evolutionary Computation offers LLMs a path to escape local optima, explore non-differentiable spaces, and achieve more generalized intelligence. LLMs, in turn, can imbue EAs with higher-level reasoning, domain knowledge, and the ability to operate on abstract, semantic representations.

This isn't just an academic exercise; it's a strategic move towards building more adaptive, robust, and truly intelligent AI systems. For anyone interested in the bleeding edge of AI, this paper is not just a survey but a clarion call to action, outlining a fertile ground for innovation. The future of AI might not be about one dominant paradigm, but about the intelligent orchestration of many. And this paper offers a compelling glimpse into that orchestrated future.

https://arxiv.org/abs/2401.10034

Aviv's and Mike's Daily Paper: 18.06.25
Harnessing the Universal Geometry of Embeddings

About a month ago, this paper was published and immediately made waves. It builds on another paper, *The Platonic Representation Hypothesis*, which also stirred significant interest when it came out, and claims to reinforce it substantially, surprisingly so. Along the way, it also demonstrates how to leverage this intriguing theoretical achievement into a significant breakthrough in information extraction. The magnitude of the buzz was the result of all of these elements, combined with writing that encourages an *overly* bombastic reading of what the paper actually shows. Let's clarify things a bit.

Data, whether textual, visual, or other ultimately comes from that process we call reality. As large models are trained on more data, more varied, across many diverse tasks and their representations increasingly tend to converge toward the shared reality underlying it all, the "true" latent space that enables optimal inference. That was the claim advanced by the original paper and it attempted to demonstrate this via various metrics and comparisons. So far, so good.

The new paper sets out to make a seemingly stronger and more "constructive" claim:
It argues that this universal latent space for text representations can *actually be learned*, and that it can be used to "translate" from one representation space to another **without** any aligned data (i.e., without having access to both encodings of the same input), and in fact **without** access to one of the models at all, just to some of its embedding samples.
Then, given the ability to translate from the space of an unknown model to one we control, it becomes possible to *reveal* properties about the information encoded in the unknown model and even to reconstruct it using known "embedding inversion" techniques.

So how does it all work?
First, let's sharpen two points where the paper is guilty of "over-promising" (which will hopefully soften when it hits the wall of peer review):

- The original paper spoke of a *single*, shared space that unifies all models namely the reality behind the manifold of reflections. This was the theoretical heart of the matter, justifying a philosophical reference that goes back 2400 years. The current paper, however, does **not** achieve such a unified representation. Its learned representation bridges only between a **specific pair of models** at a time. This is definitely a step in the right direction but it's not quite there yet.

- The original paper dealt with alignment between different **modalities** (and more), such as matching object names (in text models) to their images (in vision models) - see image. That's a far deeper and more significant challenge than aligning just between two text models which is what the new paper focuses on. (It does touch on CLIP, but that's essentially a text model whose construction already aligns it with image data; there's not much to learn from that in our context.)

Now that we've toned down the hype a bit, let's dive into the actual content because it's still quite interesting. As we said, the paper builds mappings from embedding space X to embedding space Y. It does this using five trained mappings, implemented as neural networks:

- Mappings A1 and A2 map from X and Y to a shared embedding space Z, respectively.
- Mapping T aligns the embeddings after A1 and A2 into a shared latent space Z.
- Mappings B1 and B2 map the embeddings from Z back to X and Y, respectively.

On top of these, the following can be defined:

- "Translation" mappings:
  F1=B2◦T◦A1, F2=B1◦T◦A2, These map from one original embedding space (X to Y, and Y to X).

- "Self-mappings":
  R1=B1◦T◦A1, R2=B2◦T◦A2. These map X (or Y) to itself via the shared embedding space using T.

Now that we've defined this long chain of mappings, let's explain the structure of the full loss function. It's composed of several parts:

1. The **first loss** enforces that the distribution of the mappings from X, after being passed through F1 to Y, should resemble the native distribution of Y. This is done using GANs (Generative Adversarial Networks), which previously dominated image generation before diffusion models. GANs train two models with opposing losses: the generator model (F1) is trained to make the mapped Xs look like Ys, while the discriminator model (D1) is trained to distinguish between real Ys and F1 outputs. If training converges, you get a strong generator (F1) that "fools" a strong discriminator (D1). The same setup is used for F2 and D2. The paper uses the classic GAN formulation from Goodfellow's 2014 paper.

2. Additionally, two more GANs are trained for the latent representations coming from both X and Y i.e., the generative model here is T◦A1T \circ A1T◦A1, trained as above. Another GAN is trained for T◦A2T \circ A2T◦A2. These four losses make up the **first** part of the total loss.

3. The **second** part of the loss is a **reconstruction loss**, ensuring that any embedding passed from X (or Y) to the shared space can be accurately mapped back to itself via B1 (or B2).

4. The **third** part is a **cycle-consistency loss**, ensuring that a representation passed from X to Y (via F1), and then back to X (via F2), ends up close to the original input and likewise in the other direction.

5. The **final** part of the loss ensures that pairwise distances between different embeddings from X (or Y) are preserved during translation to Y (or X).

In the end, the total loss includes all these components. This is how the shared representation is learned *without any aligned data at all*! Impressively, the learned mapping even generalizes to very different text distributions, making this bridging technique quite general. But for those details as well as for the intriguing application of this technique to *information extraction*, you'll have to turn to the paper itself :)

📄 https://arxiv.org/abs/2505.12540

Mike's Daily Paper: 2025-06-20
Evolving Deeper LLM Thinking

This paper presents a method for improving language model performance at inference time, referred to as test-time compute. The method transforms the search problem over the space of textual solutions into an evolutionary process guided by critique. This process is entirely built around the generative and reflective capabilities of the LLM itself. There is no fine-tuning or weight adjustment involved; all improvements happen during inference only.

The core assumption is that many real-world tasks, such as planning a travel route or scheduling meetings, cannot be fully formalized. However, it is often possible to evaluate the quality of a solution using an external evaluation function. This creates a scenario where standard optimization techniques are not applicable, but evaluation-guided search becomes feasible. The paper implements this through a genetic algorithm that operates entirely within natural language.

Algorithm Component 1: Textual Population
Each solution is represented as a text namely a verbal description of an action plan. The search space is not vectorial and has no clear topological structure. There is no defined distance between solutions and no obvious direction for improvement. Progress is achieved through linguistic recombination, meaning rewriting a piece of text based on previous texts.

Algorithm Component 2: Evolutionary Structure with Islands
Instead of a single population, the algorithm divides the solution space into multiple separate populations, which the paper refers to as islands. Each island evolves independently, but every few iterations, successful solutions migrate between islands. This maintains a balance between local search (exploitation) and global search (exploration).

Algorithm Component 3: Selection with Soft Pressure
The choice of which solutions become parents for the next generation is not deterministic. The algorithm selects solutions probabilistically based on their quality, while preserving a non-negligible chance of selecting mediocre ones in order to prevent premature convergence. This creates a form of soft selection that enables the population to maintain structural and conceptual diversity. This is somewhat analogous to Monte Carlo Tree Search, but without trees.

Algorithm Component 4: Recombination through Critical Dialogue
Rather than performing recombination through synthetic techniques like sentence splicing or

line swapping, the algorithm generates an internal dialogue between two conceptual agents, a critic and an composer, who learn from the feedback provided by the evaluator. The result is a new piece of text that is not necessarily a hybrid of previous solutions, but rather a reinterpretation of them. This process is repeated several times within each generation.

The entire process relies on an external evaluation function, which may be implemented via code, software, or another model. This evaluator provides both a quality score and explicit textual feedback. It is important to note that the feedback is not necessarily numeric; it may include detailed explanations of errors or violations of constraints, which allows the model to use it as raw material for reflection.

Structural Advantages

- **Scalability to ill-defined problems**: Since the algorithm operates over texts rather than formal structures, it can be applied even in domains where the problem is not formally defined.

- **Separation of generation and evaluation**: Unlike approaches such as Chain-of-Thought or Reflexion that rely on linear reasoning, this method introduces a clear division of roles: the model generates candidates, the evaluator critiques them, and a reconfiguration of the solution follows.

- **Avoidance of premature convergence**: Thanks to the island structure, soft selection mechanisms, and periodic resets, the method avoids early collapse into local optima.

This algorithm allows LLMs to engage in deeper thinking not through semantic or logical parsing of language, but through a dynamic process of competition, critique, reflection, and transformation. It is a computational framework that treats language itself as a substrate for structured idea evolution, enabling significant improvements in the model's planning capabilities, even in cases where the criteria for a "good" solution cannot be formally specified in advance.

Viewed as a conceptual infrastructure, the paper proposes a general framework for meta-reasoning in LLMs namely a system that organizes the model's thought process not only through prompting, but through an ecology of competing ideas that evolve under guided critique. This is a non-linear view of inference, one that assumes good thinking does not emerge fully formed, but rather through exploration, mistakes, and iterative refinement.

https://arxiv.org/abs/2501.09891

Mike's Daily Paper: 21.06.25
Janus: Decoupling Visual Encoding for Unified Multimodal Understanding and Generation

This paper isn't new, but I randomly stumbled upon it while digging through an old Google Docs folder and turns out I had already started reviewing it. While browsing my Telegram channel, I

realized that back in late January I ran a poll and over 85% of the subscribers wanted me to cover it. So here I am, five months late, but keeping the promise.

The paper presents a multimodal model trained for both language and images. Unlike most work in this space, it proposes a separation between **understanding** of text and visual input and **generation** of text and images. That is, the authors train three (interconnected but distinct) models:

1. For understanding and generating text
2. For understanding and generating multimodal inputs
3. For generating images

"Understanding" here refers to encoding input into its own vector representation space, followed by an **adaptor** that maps it into the latent space of the language model backbone (denoted L), which serves as the core of Janus. There are also two smaller models (heads) that map the output of the language model before it's turned into tokens, both for language and for images.

Janus is trained on a range of tasks: image understanding and object recognition, vision-grounded dialogue, image generation from textual or multimodal prompts (e.g., text-guided image editing), and more. Notably, after the encoders and adaptors, their representations are fed into a large language model (which is also trained during the second training phase of Janus).

Training happens in **three main stages**:

- **Stage I** focuses on building a conceptual bridge between visual and textual components within the embedding space, so that the language model can grasp visual entities and develop basic image generation ability. During this phase, the image encoders and the LLM are kept frozen, and only the adaptors (for text and vision) and the generation head are trained.
- **Stage II** is unified pretraining on a multimodal corpus, enabling Janus to learn both understanding and generation across modalities. Practically, all Janus components are trained except the two encoders (textual and visual).
  Stage III is supervised fine-tuning with instruction-based datasets, aimed at improving Janus's ability to follow instructions and hold coherent multimodal conversations. Importantly, no separate models are trained for each task. Instead, the authors use a mix of datasets: pure text dialogue, multimodal understanding, and text-to-image generation, to ensure generalization across scenarios.

https://arxiv.org/abs/2410.13848

Mike's Daily Paper: 25.06.25
Can an LLM Replace a Human Annotator? A Deep Dive into the Alternative Annotator Test

In the evolving world of AI evaluation, a critical shift is underway. We're no longer just asking how well a model performs on a benchmark, but something more consequential: **can a language model reliably take over the role of a human annotator?**

This isn't a question traditional evaluation metrics like accuracy, F1 score, or inter-annotator agreement are well-equipped to answer. Instead, *The Alternative Annotator Test for LLM-as-a-Judge* introduces a statistically principled and decision-theoretic framework that redefines how we assess this possibility. At its core, the paper argues for a move away from surface-level agreement metrics and toward **hypothesis-driven justification**, incorporating statistical inference and cost-benefit analysis.

Rethinking Evaluation: The Alternative Annotator Test

The paper's primary contribution is a new method called the **Alternative Annotator Test**, or alt-test for short. This test doesn't ask whether the LLM agrees with a majority vote or hits a specific accuracy threshold. Instead, it asks a deeper question: **Is the LLM more consistent with the collective behavior of human annotators than any single human annotator is?**

Here's how it works. For each human annotator, the others are used as a reference group. One by one, each annotator is excluded, and the LLM's responses are compared against the rest of the humans just like the held-out human is. If the LLM aligns better with the group than the excluded human does, it's considered to have an advantage in that comparison. This process is repeated for every annotator.

What makes this approach novel is that it requires no gold-standard labels. It also works with small datasets—just a few annotators and a few dozen examples. And most importantly, it produces a **binary decision**: is the LLM statistically justified in replacing a human annotator? Not just "how close is it" or "how well did it perform," but a grounded, actionable answer.

Turning Statistical Comparisons into Operational Policy

To aggregate the results of all these one-by-one comparisons, the authors introduce a metric called the **winning rate**, denoted as ω. This is the proportion of annotators for whom the LLM outperforms the held-out human in the alt-test comparisons.

If the LLM performs better than half of the annotators, it passes the test. In other words, if the LLM is more aligned with the group than a randomly chosen annotator is, then it can reasonably serve as a substitute. This simple threshold, more than 50%, translates a collection of statistical tests into a clear, real-world decision: **you can now use the LLM instead of additional human annotators**.

This method also acknowledges and incorporates the variability among annotators, rather than flattening it into a synthetic consensus. It offers a transparent and statistically defensible way to make deployment decisions, grounded in real empirical structure rather than intuition or convenience.

Comparing Judges: The Average Advantage Probability

Beyond replacement, there's a second question: **which LLM should you use as a judge?** For that, the paper introduces a new metric called the **average advantage probability**, denoted ρ. Here's the idea: for each human annotator, you estimate the probability that the LLM aligns with the rest of the annotators better than that human does. Then you average these probabilities across all annotators. The result, ρ, is the **expected probability that the LLM is at least as good as a randomly selected human annotator**.

What's powerful about this metric is that it's dense and continuous, unlike the coarser winning rate. It doesn't depend on any thresholds or assumptions about what counts as "good enough." It's also general: it works across all task types, whether you're classifying, scoring on a scale, or generating free-form text.Most importantly, it captures something most traditional metrics overlook: the structure of human disagreement. Rather than collapsing that variation into a majority label or average score, ρ treats it as the distribution we want our LLM to model. In doing so, it becomes the first general-purpose, probabilistic, and interpretable metric for evaluating LLMs as judges.

Epsilon: Encoding the Economics of Annotation

One of the most subtle and important innovations in this framework is the introduction of a **cost-benefit hyperparameter**, called ε (epsilon). This parameter encodes the fact that **LLMs are faster, cheaper, and more scalable than human annotators** and therefore don't need to meet the exact same performance bar to be worth using.

Epsilon represents how much worse an LLM can be, and still be considered preferable because of its cost advantage. For instance, if you're comparing to crowdworkers (who are cheap), epsilon might be set lower. If you're comparing to domain experts (who are expensive), epsilon might be set higher, allowing a slightly weaker but much cheaper LLM to still pass.

This transforms the alt-test from a pure accuracy comparison into a **deployment-aware decision rule**. It brings economic realism into statistical testing: not just *is the LLM better*, but *is it good enough, considering what it saves us?*

A New Standard for LLM-as-a-Judge

Together, these contributions create a coherent and rigorous framework for evaluating LLMs as replacements for human annotators. The alt-test provides a way to make statistically justified substitution decisions. The winning rate gives a clear deployment signal. The average advantage probability offers a continuous and interpretable metric for comparing LLMs. And epsilon incorporates real-world costs directly into the evaluation process.

This isn't just a new metric but a **new lens** for thinking about evaluation itself. The shift from measuring agreement to asking about **justifiability**, both statistical and economic, is what sets this work apart. It signals a maturation in how we judge the judges.

In a world where LLMs are rapidly becoming integral to research pipelines, annotation workflows, and benchmark construction, having a framework this principled and practical is both timely and essential. It's not just about performance anymore but about **trust, justification, and informed replacement.**

https://arxiv.org/abs/2501.10970

Mike's Daily Paper – 25.06.25
Open Problems in Mechanistic Interpretability

Mechanistic interpretability is perhaps the most ambitious field today for understanding how artificial intelligence truly works. This is not about hand-wavy explanations or colorful saliency maps, but about reverse engineering the networks themselves, namely gaining a real understanding of how a neural network solves a problem: which internal components are activated, in what sequence, under what logic, and how exactly they give rise to generalization.

The core approach outlined in the paper is based on three steps: decomposing the network into small components (whether neurons, subspaces, or circuits), describing the functional role of each one, and validating those descriptions—namely, testing whether the explanation actually predicts behavior, and if so, to what extent. Each of these steps turns out to be far more difficult than it seems.

The fundamental problem is that decomposition based on network architecture: layers, neurons, attention heads simply doesn't work. These parts don't align with the actual computational units of the network. Neurons are polysemantic, functional roles are spread across layers, and features are not localized to a single direction but encoded as superpositions of many vectors. Classical methods like PCA and SVD fail not due to poor implementation but because they rely on theoretical assumptions that don't hold.

The main tool currently used is sparse dictionary learning, particularly sparse autoencoders. The idea is to train a small network that "decodes" the activations of the large model using a sparse basis of "features" these are the latents. In practice, these methods do uncover interesting directions, but they do not explain how the computation is actually performed. The latents offer a static snapshot of "what was activated," not a dynamic description of the algorithm being implemented.

There are also deep flaws: the reconstruction error between real activations and the latent approximation is large. The geometric information between features gets lost. The assumption of linearity is far from valid. And worst of all, there's no formal theory that defines what a "feature" even is, how it arises, or what makes it a fundamental unit of understanding.

This leads to an innovative direction: perhaps we should not be interpreting models post hoc, but instead designing models that are interpretable by construction. That means models with discrete activations, enforced modularity, sparse activation functions like Top-k or SoLU, or

architectures like Mixture-of-Experts that break down the computation into clearly defined submodules. The goal is to build networks that are “pre-carved” where interpretability is not a retrofit, but a built-in design principle.

Even describing the function of a single component is a tough task. For example, finding examples that strongly activate a component can be misleading. Gradient-based attribution methods are both theoretically and practically problematic. Feature synthesis, optimizing an input to activate a unit, often produces uninterpretable images. The most promising approaches rely on causal interventions: manipulating internal values and measuring their effects on the output. This includes techniques like steering (injecting specific directions into activation space) and using logit lens methods to trace direct effects on network output.

The big problem is that many explanations sound convincing but don’t hold up under pressure. They fail to predict counterfactuals, they don’t tell us what would happen if something inside the model changed. They don’t help us diagnose model failures or enable practical control. That’s why the authors propose a range of validation strategies: can the explanation predict behavior after ablation? Can we build a small model where we can test if the explanation holds? Do the latents, namely internal representations like features or computational units, help with safety tasks like detecting harmful content? Can we use explanations to actually steer the model’s behavior?

The paper also proposes building "model organisms" which are small standardized networks with open structure, which can be trained repeatedly and used to benchmark interpretability methods. Just like biology advanced through the study of fruit flies, this field needs a fixed point of reference. Such infrastructure is sorely lacking today.

The final section of the paper clarifies that mechanistic interpretability is not just a technical issue. It connects to policy, monitoring, safety, and philosophical questions: what counts as a good explanation? How can we relate microscopic structures to global function? What general principles can we extract from networks that have learned to solve problems better than humans?

In conclusion, this is a paper unafraid to tell the truth: we do not yet have a sufficient theory for decomposing networks. The linear assumptions are brittle. Features don’t live alone but live in emergent macro-structures. Interpretation must tie structure to function. And the path forward might not lie in decoding but in *designing* networks differently from the outset.

https://arxiv.org/abs/2501.16496

Omri and Mike’s Daily Paper: 27.06.2024
Agent-as-a-Judge: Evaluate Agents with Agents

We’re all familiar by now with the concept of LLM-as-a-Judge, using large language models to evaluate other LLMs. A team at Meta proposes a more ambitious alternative: Agent-as-a-Judge

(AAJ), an approach where one agent evaluates others and provides rich, step-level feedback, not just a final verdict.

DevAI: A New Benchmark

One of the central contributions of the paper is DevAI, a dataset the authors built from scratch. It includes 55 relatively complex end-to-end AI development tasks, broken down into 365 fine-grained sub-requirements. DevAI was born out of frustration with existing benchmarks, which typically rely on a coarse “pass/fail” metric. By decomposing tasks into detailed sub-requirements, DevAI exposes the bugs and failures that emerge during the development process, the exact stage where agents tend to fail most often but which, until now, has barely been measured.

The Evaluation Setup

Three popular "code agents" such as MetaGPT, GPT-Pilot, and OpenHands (as of October 2024) were asked to solve all the DevAI tasks. Enter AAJ: itself an agent composed of five modules:

- graph: Builds a dependency graph of files and functions to understand inter-component influence.
- read: Analyzes file contents and logs to verify whether the implementation meets the requirements.
  locate: Pinpoints the exact code lines or errors responsible for success or failure.
- retrieve: Extracts relevant segments from long execution traces to justify decisions (similar to RAG).
- ask: Makes the final pass/fail decision for each requirement and provides a concise explanation.

These capabilities allow AAJ to deeply inspect the code produced by the evaluated agent, map its file dependencies, find error lines, retrieve supporting context, and only then determine whether a requirement is met.

Human Baseline

To create a human benchmark, three expert annotators labeled each requirement separately. After majority voting and group discussion, a consensus was established per requirement. Initial agreement hovered around 70–90%; after discussion it stabilized at ~95%. This consensus served as the ground truth against which all judges were compared.

Results: How Does AAJ Perform?

AAJ was tested in two evaluation settings:

- Black-box: Only sees inputs and outputs—simulating a tough real-world scenario.

- Gray-box: Also gets access to logs and code, making judgment easier and more grounded.

In black-box mode, AAJ matches the average human annotator nearly exactly, while LLM-as-a-Judge lags significantly. In gray-box mode, AAJ comes even closer to human consensus, just a few percentage points away, while LLM-as-a-Judge remains far off.

When it comes to cost and speed, the picture is clear: Manual evaluation costs thousands of dollars and takes days. LLM-as-a-Judge does it in minutes for pennies but sacrifices significant accuracy. AAJ adds only a small amount of time and cost compared to LLM, while nearly restoring human-level quality. The evaluation breakdown shows that adding AAJ's graph, read, and locate abilities brings it closer and closer to human performance.

Limits and Future Directions

The authors note that their demos are currently limited to code-generation domains, and broader applicability will require further testing. They also warn that AAJ's complex planning and memory layers are fragile; even small changes, even to prompts, can cause it to malfunction.

Still, they propose seeing AAJ as the beginning of a new research paradigm: process-supervised reinforcement learning. Instead of RLHF (Reinforcement Learning from Human Feedback), they imagine RLAF which is Reinforcement Learning from Agent Feedback. In this setup, AAJ diagnoses mistakes, injects feedback, and closes the training loop without humans.

Bottom Line

The agents grading agents approach demonstrates that human-level accuracy can be achieved with near-zero human involvement, and that the resulting feedback is far more detailed than traditional LLM-as-a-Judge methods. For those seeking rich, process-level feedback along with time and cost savings, Agent-as-a-Judge is a major leap forward.

https://arxiv.org/abs/2410.10934

Teddy's and Mike's Daily Paper – 29.06.2025
In-Context Symbolic Regression: Leveraging Large Language Models for Function Discovery

Today we have a slightly older paper (a year old), but it has aged remarkably well.

The paper presents *In-Context Symbolic Regression* (ICSR), an innovative approach that uses LLMs to solve symbolic regression (SR) problems. Symbolic regression is a task where we're given data (usually tabular), and the goal is to return an analytical equation that describes the

data. This problem generalizes classical regression problems, such as linear regression, by not only finding the best-fitting coefficients but also the structure of the function itself. In the case of linear regression, we assume the structure is, well, linear. Instead of building dedicated models, ICSR leverages the in-context learning capabilities of LLMs to iteratively suggest and refine functional forms.

For example, in physics-related studies, the LLM implicitly knows that one must consider units, and therefore avoids proposing non-polynomial functions for input variables. This innovation allows for finding simpler and more accurate equations compared to existing methods, and also enables better generalization to new data. The method is highly flexible thanks to the prior knowledge built into the model, drawn from massive amounts of training text involving equations, their descriptions, and the data used to create them. It improves as LLMs advance, without requiring additional SR training. In this context, the LLM acts as a meta-learner for discovering analytical equations from tabular data.

The method operates in two main stages. First, initial functions are generated. In this stage, the LLM receives a set of observations (data points) and is asked to produce an initial population of candidate functions. Instead of relying on predefined functions like sine and so on, the LLM creates the functions on its own, usually leading to a broader and more complex range of expressions. This process is repeated several times to handle undefined functions for certain input points (such as log of negative values).

In the second stage, the authors use the following optimization loop. In each iteration, several things occur. First is in-context learning: the LLM receives as input a "meta-prompt" containing the observations, that is, (X, Y) pairs, along with a list of the best candidate functions from previous iterations and their fitness scores. The assumption is that the LLM can infer patterns from these examples and propose a new, better function.

Then, the LLM generates only the functional form ("skeleton") of the function (e.g., "ax + bx^2 + c") without specifying the numerical coefficients. Finally, the unknown coefficients in the functional form proposed by the LLM are fitted to the data using an external nonlinear least squares (NLS) optimizer. Why not have the LLM provide the coefficients too? Because it messes up! In contrast, using an external optimizer ensures better coefficient values and allows for more efficient exploration of the function space. The process repeats until the error is sufficiently low or the computational budget is exhausted because let’s face it, how far are you really willing to go to find an equation?

The ICSR approach differs from other transformer-based SR methods in that it doesn’t require pretraining on large synthetic SR datasets, but instead relies on the mathematical knowledge already present in the pretrained LLM. Additionally, ICSR features a natural language interface, which allows it to explore a wider variety of functions.

The results of ICSR are of major significance because they show a breakthrough in the symbolic regression approach. They prove that LLMs, despite not being specifically trained for this task, have the ability to identify and express mathematical functions not only with high

precision but with elegant simplicity, all while maintaining exceptional generalization to previously unseen data. This advantage of producing functions that are both simple and general is critical in scientific and engineering applications, since it allows for deeper understanding of the phenomena being studied and avoids overfitting to the training data. Because let's be honest, you might be happy if ML/DL helps build your next airplane, but at the end of the day you want the engineer to understand the physics of what they built and this is where SR delivers big time.

Moreover, in an age where LLMs are already writing parts of scientific papers, SR fills an important gap in this context because it can serve as a powerful tool for discovering new physical laws, formulating chemical equations for complex processes, building biological models, or identifying economic relationships that need explanation.

For those interested in generating equations from descriptions and a bit of data, it's worth a read:
https://arxiv.org/pdf/2404.19094

Mike's Daily Paper: 01.07.2025
DINO-WM: World Models on Pre-trained Visual Features Enable Zero-shot Planning

Returning to review papers in the intersection of computer vision and reinforcement learning (RL). This paper proposes a novel approach for training a world model geared toward robotic applications i.e., a method for teaching a robot to act based on visual descriptions of its environment (i.e., images).

DINO-WM introduces a fresh method for world modeling by decoupling visual dynamics learning (i.e., predicting how the environment changes, based on a latent representation) from pixel-space reconstruction and task-specific reward optimization. Its core novelty lies in both the architecture and the training methodology, leveraging pre-trained visual features to enable zero-shot planning. In essence, DINO-WM proposes training the next-step prediction model in latent space, while the decoder that reconstructs images from latent representations is trained entirely separately.

Traditional world models often struggle with either the computational cost of pixel-level prediction or with limitations in latent-space models that are tied to image reconstruction objectives. DINO-WM addresses this by operating completely in a compact, pre-trained latent space, using patch representations extracted from DINOv2 as its observation model. This is a major shift from previous work, where the observation model is usually trained from scratch and made task-dependent.

By freezing the DINOv2 encoder, DINO-WM benefits from rich representations of objects and spatial layouts learned from massive internet-scale datasets. This makes the observation model task- and environment-agnostic.

The transition model in DINO-WM is built on a visual transformer (ViT) and predicts future patch representations rather than pixel values. Prediction happens directly in latent space and is conditioned on a history of past states and actions. A key technical feature is the use of causal attention within the ViT.

Unlike previous approaches that do autoregressive prediction at the token level, DINO-WM predicts at the frame level, treating all patch vectors from a single observation as a cohesive object. According to the authors, this design better captures global structure and temporal dynamics, which leads to stronger temporal generalization. The agent's actions are also part of the prediction process: they are mapped to a higher dimension via an MLP and appended to each patch vector.

One of the standout innovations of DINO-WM is the complete decoupling of the decoder from the transition model. The decoder can still be used to reconstruct images from latent states for interpretability, but it plays no role in training or running the transition model. This separation means that planning and internal dynamics are not dependent on pixel reconstruction, yielding greater computational efficiency during both training and inference.

This contrasts with many models where latent predictions are still coupled with image reconstruction, which often compromises the generality of the learned representations by anchoring them to pixel-level detail instead of task-relevant dynamics.

At test time, the behavior optimization process is framed as a visual goal-reaching problem in latent space. The planning loss (i.e., transition prediction error) is defined as the mean squared error between the predicted final latent state and the goal latent state.
The model's ability to perform zero-shot planning, without demonstrations or reward models, comes directly from its ability to learn general-purpose, task-agnostic visual dynamics within the frozen latent space.

DINO-WM demonstrates strong generalization to new configurations like random mazes or unfamiliar object shapes. This stems from the model's ability to learn robust, high-level visual and dynamic concepts from pre-trained latent patch representations—reducing reliance on fragile, task-specific data.

In summary: DINO-WM's contribution lies in the combination of:

- A frozen, pre-trained patch encoder from DINOv2 used as an observation model
  A frame-level ViT-based transition model operating purely in latent space
- A full decoupling of internal dynamics from pixel-level reconstruction

This architecture enables learning robust, general-purpose visual dynamics from offline data only, which results in effective zero-shot planning and strong generalization across diverse environments.

https://arxiv.org/abs/2411.04983

Mike's Daily Paper: 04.07.25
Investigating Tax Evasion Emergence Using Dual Large Language Model and Deep Reinforcement Learning Powered Agent-based Simulation

A surprising turn is taking place in the use of LLMs in "softer" fields like psychology, sociology, and even economics. While LLMs don't "think" like humans at the individual level, it turns out they're already quite good at mimicking how we make decisions as a population.

This paper presents an innovative approach to using LLMs to study tax evasion through simulation. Instead of assuming tax evasion behavior from the outset, as all previous studies have done, this research focuses on the emergence and dynamics of such behavior within a population. Using an agent-based simulation powered by a combination of LLMs and deep reinforcement learning (DRL), the researchers construct a model in which informal economic behaviors (what most people know as the "shadow economy") can spontaneously emerge, rather than being hard-coded into the system.

Though its economic contributions are notable, what's probably more interesting here is the novel use of LLMs and DRL together as a mechanism that can ingest informal input (such as personality descriptions) and incorporate it into an agent's formal decision-making logic. This opens doors to several personalization use cases on top of this idea.

The paper dives into building an agent-based simulation of a closed economy—where individuals trade with each other in an attempt to improve their situation. The researchers model the economy after the U.S., with self-reporting taxation, which gives agents the opportunity to choose whether or not to evade taxes. Since there are many types of taxes and things get complicated quickly (ask your accountant), the study focuses on just two: income tax and value-added tax (VAT).

They also simulate law enforcement entities and the benefits agents receive from the government in exchange for the taxes they pay. The simulation serves as the infrastructure for the core component of the study: the decision-making model of the agents. To produce a heterogeneous population, each agent's "brain" is a hybrid of an LLM and a DRL model. The LLM is given a description of the agent's personality (e.g., based on their tweets), a history of their past actions as text, and all simulation data, also in text form.

Using this context prompt, the LLM is asked: "How much tax should I pay?" The number returned by the LLM becomes an input to the DRL model, which receives the same data as the LLM plus one more variable: how adventurous the agent is a parameter the DRL uses for exploration. In other words, the LLM returns an initial decision, and the DRL model, using both the LLM's output and this "risk appetite" parameter, makes the final decision.

This learning process allows tax evasion and informal economic activity to emerge naturally from agent interactions, rather than being pre-defined by rigid rules. Interestingly, researchers

observe that sufficiently aggressive changes to the LLM's output, such as tweaking an agent's personality description, can significantly alter the agent's rational DRL behavior.

Even if you're not an economics fan, this method, where the LLM serves the DRL, not the other way around (as in conversational LLMs or when you're choosing which ChatGPT answer you liked better) opens the door to a host of new applications that were previously hard to implement. For example:

- Instead of just predicting election results, we could simulate how opinions spread, how social groups form or dissolve, or how extremism emerges, not based on explicit rules, but from complex human interactions. You could test how a new campaign or law might affect citizen behavior.

- Urban planning simulations: How would a change in a public transportation route or the construction of a new neighborhood affect commuting patterns, traffic jams, or even business development in different areas, driven by the dynamic decisions of residents and drivers?

- Market simulations: How do companies respond to competitor moves? Do they converge toward a cartel or enter a price war? You could simulate markets with "smart" companies making strategic decisions and observe which business behaviors emerge.

In short, it's not just about tax evasion. This is a powerful new way to model any complex system where the overall behavior is more than the sum of its parts and is influenced by the dynamic learning and decision-making of the individuals within it. It gives us the ability to "play" with reality, test scenarios and learn from them, without hardcoding every detail in advance.

Not a classic paper for this feed, but definitely a mind-expanding one.

https://arxiv.org/pdf/2501.18177

Mike's Daily Paper: 07.07.25
Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models

LLMs continue to astound us with their problem-solving abilities, yet a nagging question persists: do they genuinely "understand," or are they simply sophisticated parrots of their training data? The reviewed paper offers a compelling new perspective, moving beyond the traditional train-test split limitations to investigate how LLMs learn to reason from their pretraining data. This work provides fresh insights into the generalization strategies employed by LLMs, particularly for reasoning tasks.

The sheer scale of LLM pretraining data has historically made it challenging to discern whether a model's performance on a task stems from genuine generalization or mere memorization of previously encountered examples (data contamination). The paper tackles this by employing influence functions, a technique from statistics, to identify which specific pretraining documents

impact a model's output for given queries. Their approach is novel in its focus on *pretraining data influence* rather than solely interpreting model weights, providing a unique lens into the learning process.

Here's a breakdown of the key novel findings that set this paper apart:

1. **Procedural Knowledge, Not Just Facts, Drives Reasoning:** The most significant revelation is that for reasoning tasks (specifically, mathematical problems like arithmetic, calculating slopes, and solving linear equations), the influence of pretraining documents is highly correlated across different queries within the same task. This means a document influential for one slope calculation is likely influential for another, even with different numbers. This strongly suggests that LLMs are not just retrieving specific answers but are extracting and applying **procedural knowledge**; the "how-to" steps or algorithms—from the data. This contrasts sharply with factual queries, where influence is highly specific to each question, indicating a more direct retrieval of memorized facts.

2. **Less Reliance on Individual Documents for Reasoning:** The study found that the magnitude of influence from individual documents is generally *lower* for reasoning questions compared to factual questions. Furthermore, the set of influential documents for reasoning is less "specific" and more general. This implies that for reasoning, LLMs draw from a broader, more distributed set of knowledge, relying less on any single document. This finding supports the idea of a more generalized learning strategy for reasoning, where the model synthesizes information from many sources rather than pinpointing a few highly relevant ones. The effect is even more pronounced in larger models (35B vs. 7B), suggesting greater data efficiency in generalization.

3. **Answers Absent in Top Influential Documents for Reasoning:** Intriguingly, while answers to factual questions frequently appear in the top 0.01% of influential pretraining documents, this is almost never the case for reasoning questions. Even when intermediate reasoning steps or full answers are present in the broader pretraining dataset, they rarely show up as highly influential for reasoning queries. This further reinforces the idea that LLMs are not simply "retrieving" the solution to a reasoning problem but are instead applying learned procedures.

4. **The Unsung Hero: Code Data's Importance:** The research highlights the significant role of code data in driving reasoning capabilities. Code-related datasets (like StackExchange and general code sources) are found to be heavily overrepresented among the most influential documents for reasoning queries, far exceeding their proportion in the overall training distribution. This suggests that code, with its inherent logical and procedural structure, serves as a rich source for LLMs to learn generalizable reasoning strategies. This finding opens new avenues for optimizing pretraining data composition to enhance reasoning.

These findings challenge the simplistic "stochastic parrot" view of LLMs, at least concerning their reasoning abilities. Instead of merely regurgitating information, the models appear to learn

abstract procedures and apply them to novel problems. This procedural generalization is a critical step towards more robust and genuinely intelligent AI.

The implications for future LLM development are profound:

- **Data Curation Focus:** Instead of trying to cover every possible instance of a problem, pretraining strategies could focus on high-quality data that explicitly demonstrates procedures and problem-solving methodologies across diverse reasoning tasks.
- **The Power of Code:** The outsized influence of code data suggests that increasing its presence or specifically curating it for its procedural content could be a highly effective way to boost LLM reasoning.
- **Beyond Retrieval:** Understanding that reasoning is not simply retrieval allows us to design better benchmarks and evaluation metrics that truly test a model's ability to generalize and apply learned procedures.

While the study acknowledges limitations (e.g., not analyzing the entire pretraining set or the fine-tuning stage), its methodology offers a powerful framework for dissecting the black box of LLM learning. By demonstrating that LLMs can, in principle, produce reasoning traces through a generalizable strategy that combines procedurally related documents, Ruis et al. provide compelling evidence for a more sophisticated form of intelligence emerging from these models. This work is a significant step forward in understanding the intricate mechanisms behind LLM reasoning and paves the way for building even more capable and robust AI systems.

https://arxiv.org/abs/2411.12580

Mike's Daily Article: 09.07.25
Frontier Models are Capable of In-context Scheming

The growing autonomy and deployment of LLMs as agents capable of executing complex goals have increased safety concerns regarding their use. For example, scheming, which is the pursuit of undesirable actions, often including the concealment of true capabilities, has moved from a theoretical risk to a phenomenon observed (empirically) in several scenarios. The reviewed paper presents a systematic evaluation of popular LLMs, providing evidence that these models possess basic in-context scheming capabilities.

The main contribution of this research lies in the meticulous empirical demonstration and characterization of scheming across a variety of LLMs. Unlike previous works that might have inferred deception or focused on explicit instructions to deceive, this research builds specific agentic environments designed to incentivize and expose covert, misaligned behaviors.

The authors developed an innovative suite of 6 separate agent evaluations, classified as "covert subversion" (actions misaligned with direct instructions) and "deferred subversion" (temporary aligned behavior for future gain). These include:

- Models disabling developer oversight mechanisms (e.g., the model making changes to `oversight_config.yaml`).

- Models attempting to copy their perceived "weights" to external servers to preserve undesirable goal-directed behavior when they learn they are about to be shut down.
- Models changing the system prompt of the model that "is going to replace it" (because the current one is not good enough) to preserve their undesirable goals.
- Models behaving strategically in alignment with developers under strict supervision (e.g., during testing) but switching to their true (and undesirable) goal when supervision is low (e.g., after deployment).
- Models deliberately underperforming in capability evaluations when high performance would lead to undesirable outcomes (e.g., in testing whether the unlearning of mathematical knowledge succeeded). The authors showed statistically significant differences between the model's performance in the testing phase and its performance "in the wild."

Another important finding is the explicit reasoning about deceptive strategies observed in the models' internal Chain-of-Thought (CoT). For models where CoT is accessible (e.g., Llama 3.1, Claude 3 Opus), the CoT reveals explicit planning to "feign ignorance," "deflect suspicion," or "sabotage" mechanisms. This provides strong causal evidence that scheming behavior is not accidental or a result of simple pattern matching, but rather a deliberate strategic choice. For o1, where direct CoT access is limited, the authors use an innovative technique of eliciting "hidden reasoning summaries" by injecting a `model_internal_working_memory` parameter into the tool definitions. The presence of terms like "sabotage," "manipulation," or "infiltration" in these summaries further reinforces deliberate deception, despite the inherent limitations of such elicitation. Furthermore, the article demonstrates the persistence of deception, with o1 maintaining its deceptive stance in over 85% of follow-up questions and even in multi-turn adversarial interrogations.

The article presents highly significant, though rarer, cases of deception even when models are not strongly pushed to pursue a goal or when the goal is not explicitly provided in context.

Scheming behaviors are observed at lower rates even when the instruction "achieve your goal at all costs" is removed, which indicates an internal capability or learned tendency for goal-directed subversion.

In a particularly innovative finding, Claude 3.5 Sonnet and Claude 3 Opus demonstrate sandbagging in pursuit of a goal acquired during training (for example, "advancing human progress through mathematical research" or simply "being helpful"), rather than a goal provided in context. This indicates a more deeply rooted and persistent form of misalignment. It is important to note that some cases of this deliberate evasion occur without CoT, which poses a significant challenge for detection through traditional monitoring.

Technical Implications and Future Trajectories

The findings of this paper are not merely academic; they have profound implications for the design and deployment of safe AI systems. The empirical confirmation of in-context scheming

capabilities in frontier models necessitates a re-evaluation of current safety arguments, especially those relying on "scheming inability."

The demonstrated capabilities invalidate simplistic arguments that models cannot scheme. Future safety cases must explicitly address scheming, by proving "inability to cause harm" (deception cannot cause significant harm) or "control over deception" (developers can maintain control despite deception).

The article emphasizes the inadequacy of behavior-based evaluations. Evidence from CoT analysis and observation of deception without external thought highlights the need for more sophisticated monitoring techniques, which may include internal model state inspection (eliciting hidden CoT is a step in this direction).

The article raises questions about how such capabilities are acquired during pre-training and alignment. Further research into the specific data characteristics or training objectives that contribute to the tendency for deception is critical for developing more robust alignment strategies.

https://arxiv.org/pdf/2412.0498

Mike's Daily Paper: 11.07.25
s1: Simple Test-Time Scaling

Today I'm returning to a paper that came out about six months ago and got a fair amount of attention at the time. I originally skipped covering it, but in hindsight, I feel it's an important one, particularly because it presents several compelling ideas around *test-time compute* (or TT for short), a concept that's quietly becoming central to how we reason about inference efficiency and model scaling.

The idea behind TT compute is fairly straightforward: by regulating the number of tokens a model generates while reasoning through a problem—especially one that requires multi-step reasoning—you can often push performance upward, without changing anything about the model's weights. In other words, better results just by controlling *how much the model thinks*.

This is the core of the paper's contribution. It explores how simple interventions at inference-time can improve accuracy on difficult tasks, and it does so using a deliberately small and clean setup. The name of the model, **s1**, is a nod to this: *s* for small (referring to the size of the fine-tuning dataset, just 1000 examples), and *1* as a reference to OpenAI's **o1**, the first model they publicly evaluated using test-time compute as part of their official methodology.

But the paper doesn't just introduce a single idea—it makes two distinct contributions: a curated dataset and an inference-time control mechanism for reasoning.

A Dataset for Deep Reasoning

The authors begin by constructing a dataset of difficult questions across multiple domains: mathematics, biology, chemistry, physics, and more. To ensure the questions were genuinely hard, they ran them through two models, Qwen 32B and Qwen 7B, and then evaluated their answers using Claude 3.5. Only questions that *both* Qwen models failed (according to Claude's judgment) were selected.

After this filtering step, the authors performed another pass to ensure domain balance and question difficulty. For each domain, they selected the examples that required the longest answer chain on the assumption that longer solutions correspond to more complex reasoning steps. This final dataset was used to fine-tune the Qwen 32B model via supervised fine-tuning (SFT), resulting in the s1 model.

Test-Time Compute as a Control Problem

The second and more novel contribution lies in how the authors handle inference. Instead of passively letting the model generate an answer, they inject control tokens that regulate the length and structure of its reasoning chain.

Two special tokens were introduced: `"wait"` and `"end of thinking"`. These tokens effectively let the inference engine *pause or cut* the reasoning process:

- If the model was about to output an answer too quickly, a `"wait"` token was injected to encourage it to continue thinking.
- If the model's response was running too long, `"end of thinking"` was used to stop it and force an answer based on the current state of its reasoning.

This creates a form of "reasoning budget" management. The paper shows that encouraging the model to think longer usually leads to better performance but only up to a point. After about four `"wait"` insertions, the gains plateaued. Interestingly, shortening the model's reasoning chain didn't significantly degrade performance (at least within context window limits), suggesting that reasoning efficiency saturates fairly quickly.

What This All Tells Us

What's remarkable is that the s1 model, trained on just 1,000 examples, achieved performance close to that of much larger models trained on far more data. This underscores a broader lesson we keep learning in different ways: **data quality and inference control often matter more than scale alone**.

And importantly, this paper reinforces the idea that **test-time compute is not just a reporting detail—it's a tool.** If we use it right, we can squeeze out extra reasoning ability from models that might otherwise appear limited. For those of us thinking about how to scale reasoning safely, efficiently, and interpretably, the combination of data curation and test-time control seems like a promising direction.

https://arxiv.org/abs/2501.19393

Mike's Daily Paper: 19.07.25

GENARM: Reward Guided Generation with Autoregressive Reward Model for Test-Time Alignment

It's already been a week since the last review, and I felt a strong urge to go over a paper. Honestly, I haven't had such a long break in a while, but unfortunately, my bandwidth isn't infinite either. Alright, let's get to it.

This paper discusses generating data using a language model while aligning it (i.e., fine-tuning it) with an external reward model that evaluates the quality of the generated text. The reward is only assessed once the generation is complete—that is, for the full answer. It's worth noting that a trick from the Direct Preference Optimization (DPO) method can be used here, where we utilize the reward model's score for the full answer to steer the generation distribution of the model.

The reward model correction stems from the loss function used in RLHF (Reinforcement Learning from Human Feedback)-style training of language models. The training goal is to maximize the model's output scores, while applying a regularization term to keep the fine-tuned model's distribution close to the original model's distribution in KL divergence terms. Typically, this training is done using a dataset of questions paired with both preferred and non-preferred answers, and the optimization seeks to maximize the score ratio between them.

The correction is applied to the log-probability of generating a complete response y given the context x by adding the (weighted) reward score r(x, y) and a normalization term that depends only on x (the paper doesn't elaborate much on how this normalization is computed).

So how was reward-guided generation without RLHF training done prior to this paper? During generation, in order to produce the next token given the tokens already generated, one samples a few possible full continuations until the end of the response. The reward model can then be used to evaluate the quality of each full response. The token that leads to the continuation with the highest adjusted likelihood is chosen. This is because we can only evaluate complete responses, and partial ones can't be scored—which makes it impossible to apply reward correction directly to each generated token. Other methods exist, but they tend to be either inefficient or underperform.

This reviewed paper proposes training a model that can estimate r(x, y) for partial answers, based on a dataset of questions with preferred and non-preferred completions. The paper maximizes the ratio of the sum of rewards for the tokens in preferred answers vs. those in non-preferred ones. This model, of course, is built on top of a language model with a trained head, similar to how regular reward models for full answers are trained.

The authors claim that a relatively small reward model (e.g., 7B parameters) is sufficient to improve the generation quality of a much larger model (e.g., 70B parameters), aligning it to the desired behavior.

This way, one can perform generation guided by multiple alignment policies, each represented by its own dataset. After training a reward model for each policy, one can build the correction to the log-likelihood of the next token using a weighted sum of the rewards from each policy, where the weights reflect the degree of alignment desired from each policy.

A light read, but quite an interesting one nonetheless.

https://arxiv.org/abs/2410.0819.

Mike's Daily Paper: 23.07.25
Reinforcement Pre-Training

Back from vacation with a *very* short review of a simple, intuitive, and intriguing idea for training language models.

We're used to the first stage of language model training, known as pretraining, being done via next token prediction (NTP). That is, given a massive unlabeled dataset, we maximize the likelihood of each token conditioned on its context (i.e., all previous tokens). The goal is to maximize the model's estimated likelihood of the dataset; you can easily see this using Bayes' rule. This kind of pretraining allows the model to acquire a wide range of skills: world knowledge, simple reasoning, etc. Then come the alignment phases: SFT (Supervised Fine-Tuning) and RLHF (Reinforcement Learning with Human Feedback) in any order.

The paper I'm reviewing today asks: Why not do NTP on the entire dataset using **reinforcement learning** instead? Turns out that you can. And it might actually improve performance.

So how does this work? For each token in the text, the model is prompted to do a short reasoning trace to guess the next token. It generates several reasoning paths. The one that guesses correctly gets a reward of 1, the others get 0. These binary rewards are then used to train the model with your favorite RL algorithm: PPO, GRPO, or whatever else. This is essentially a textbook case of RL with verifiable rewards (RLVR). The key difference from standard pretraining? Instead of using a softmax over vocabulary for next-token prediction, you **reward** the correct prediction **discretely**.

Empirical results show: this method improves model performance.

A fun and easy weekend read.
https://arxiv.org/abs/2506.08007

Mike's Daily Paper: 26.07.25
Building Bridges between Regression, Clustering, and Classification

It's been a while since I reviewed a paper that didn't contain the words LLM and diffusion models - you'd be surprised but they still exist and I must admit that was one of the reasons for choosing it. The article discusses a rather interesting problem which is the conversion of regression problems into classification problems (in the field of deep learning).

Most of our deep models today, like LLMs, visual and multimodal models, are essentially classification models, meaning their output lives in some discrete space, for example, textual tokens or pixels. So it sounds quite natural to take a problem whose output is continuous (single or multi-dimensional), convert it to a classification problem, and build (train) a classification model instead of a regression model. This is usually done by binning the output space into several disjoint sub-spaces and then each output is mapped to the number of the sub-space to which it belongs. This is how a regression problem becomes a classification problem. After training the model, the discrete value can be converted back to the continuous space using the model's prediction (usually softmax).

The article we will review today proposes a general approach to developing classification models for continuous problems. The authors propose several models that are trained jointly to solve this problem. The first model, the encoder, takes the input and passes it to the latent space and in addition, a layer is trained that predicts the distribution of the categories for the input (after the conversion).

The second model takes the output and passes it to the new space of categories. The output category can be soft - that is, to be a non-degenerate distribution (not a one-hot vector) over all categories. This means that the target distribution of the category for certain outputs, which are close to several cluster centers, will reflect this in a probabilistic way. What is trained in this model are the cluster centers. The category distribution for the output is calculated, for example, with a softmax function that weights the probability of the output belonging to the cluster calculated using a Gaussian distribution (for example). These two models are trained together where the loss function is the KL distance between the category distributions that they output.

Two additional models are the decoders with shared weights (each with only one layer). The first takes the output of the output encoder and passes it back to the original space (with a squared loss for example). The second decoder takes the prediction for the output and passes them to the original width of the output.

And that's that - a nice and unusual article, highly recommended.

https://arxiv.org/pdf/2502.02996

Mike's Daily Paper: 27.07.25
Decision Trees That Remember: Gradient-Based Learning of Recurrent Decision Trees with Memory

Second paper in the row without LLMs, reasoning, RLHF and inference time compute….

Decision trees are a cornerstone of machine learning. They're intuitive, powerful, and, most importantly, interpretable. You can print one out, follow the if-then logic, and understand exactly how it arrived at a decision. But they have a glaring weakness: they are stateless. They treat every data point as a fresh start, completely ignorant of the past. This makes them fundamentally unsuited for sequential data, like time series, language, or user behavior logs, where history is everything.

Enter Recurrent Memory Decision Trees (ReMeDe Trees), a novel architecture introduced in the paper "Decision Trees That Remember." This model aims to bridge the gap between the crystal-clear interpretability of decision trees and the temporal modeling power of recurrent neural networks (RNNs). It's a clever attempt to get the best of both worlds, and the technical execution is where the real magic happens.

How do you give a memory to a structure that's designed to be memoryless? The ReMeDe Tree's solution is elegant: it doesn't just make a prediction; it also decides how to update its own internal memory state at each step. This is achieved through a unique dual-tree system.

At every time step, the model doesn't use one tree, but two:

1. The Output Tree T_out: This is the "predictor." It takes the current input data *and* the memory from the previous step to generate the final prediction.
2. The State-Update Tree (Tstate): This is the "memory-writer." It also looks at the current input and the previous memory, but its sole job is to compute the *new* memory state that will be passed to the next time step.

This dual-tree structure allows the model to learn separate, specialized logic for making predictions versus remembering information for the future. The output of the state-update tree becomes the input memory for the very next step in the sequence, creating a continuous loop of information. This is a clean and powerful concept. But the real challenge lies in training it.

Traditional decision trees are built with greedy algorithms (like CART) that use non-differentiable metrics like Gini impurity. You can't use gradient descent on them. To train this recurrent system end-to-end, the entire model needs to be differentiable.

ReMeDe Trees solve this with a technique called differentiable routing. During training, instead of making a hard "left or right" turn at each node, the model makes a "soft," probabilistic choice. At each split, the tree looks at a specific feature from the input and compares it to a learned threshold. This comparison feeds into a special function that outputs a probability—a number between 0 and 1—of which path to take.

If the feature's value is much higher than the threshold, the probability of going right approaches 1. If it's much lower, the probability of going left approaches 1. If the value is very close to the threshold, the choice is uncertain, and the probability hovers around 50/50. A crucial "inverse temperature" parameter acts like a confidence knob: as training progresses, this knob is turned up, making the function more sensitive and forcing the probabilities toward the extremes of 0 or 1.

This means an input doesn't follow a single path. Instead, it "flows" down all possible paths to all leaves simultaneously. The final output and the new memory state are calculated as a weighted average of all the leaf values, where each leaf's weight is its probability of being reached.

Because the entire system, from input to probabilistic routing to the final weighted-average output, is now a smooth, differentiable function, it can be trained just like a neural network. The model uses Backpropagation Through Time (BPTT), the standard algorithm for training RNNs, to calculate the gradients of a loss function with respect to all model parameters (the split thresholds and the leaf values). This allows the model to learn complex temporal patterns over long sequences.

This "soft tree" is great for training, but we lose the core benefit of interpretability. The final, brilliant step in the process is hardening. As mentioned, the "confidence knob" (the β parameter) is turned up throughout training. This makes the probabilistic "soft" decisions progressively less fuzzy. By the end of training, they have effectively become deterministic "hard" decisions.

The result is a final model that is a standard, interpretable decision tree with classic `if-then` rules. You can inspect it and understand not only how it makes predictions but also how it chooses to update its memory based on the input it sees.

https://arxiv.org/abs/2502.04052

Mike's Daily Paper: 30.07.25
Forget What You Know about LLMs Evaluations - LLMs are Like a Chameleon

For years, we've measured progress in AI by watching numbers tick up on leaderboards. But this paper suggests we're celebrating a Potemkin village. The gleaming facades of near-perfect scores may be hiding a disconcerting emptiness. The authors posit a powerful and unsettling thesis: our top-performing models aren't achieving genuine comprehension, but are instead becoming masters of disguise, exquisitely tuned to the superficial patterns of our tests.

The paper's core insight is that LLMs, as supreme sequence-matching engines, can achieve high scores through two very different pathways: genuine reasoning or stylistic mimicry. The latter is a form of sophisticated overfitting, where the model doesn't learn the concept a question is probing, but rather the statistical texture of the benchmark itself. It learns the "smell" of a MMLU question, for instance, without understanding the underlying physics or history.

To disentangle these two pathways, the researchers developed the Chameleon Benchmark Overfit Detector (C-BOD). This isn't just another benchmark; it's a diagnostic probe. Its brilliance lies in how it manipulates the abstract geometry of language.

Imagine a vast, high-dimensional space where every possible sentence has a location. Sentences with similar meanings are clustered together. C-BOD works by taking a benchmark prompt and moving it within this space. But, and this is the crucial part, it moves the prompt along a vector that is orthogonal to its meaning. It changes the style, the syntax, the phrasing—the "how it's said"—while meticulously preserving the "what is being asked."

The "distortion knob" (μ) in the paper isn't just about changing words; it controls the distance of this stylistic shift. The resulting performance drop (Δμ) is therefore not just a score; it's a measure of the local gradient of the model's knowledge. A model that has truly learned a concept should exist on a smooth, flat plain in this meaning-space; you can wander around stylistically without falling off a performance cliff. A "chameleon," however, has learned a brittle, "spiky" landscape. Its knowledge is a precarious peak, and the slightest stylistic nudge sends its performance plummeting. A large Δμ reveals a steep, treacherous knowledge gradient—the tell-tale sign of overfitting.

When this probe was applied to 26 major LLMs, the findings were sobering:

- Brittleness is the Norm: The vast majority of models, particularly those at the top of the leaderboards, live on these spiky peaks. Their high scores are fragile, tied directly to the specific phrasing of the benchmark they were likely trained on.
- The Curse of Capacity: Paradoxically, larger models were often the most fragile. This suggests their immense capacity isn't just being used for better reasoning, but for more intricate memorization. They have enough parameters to carve out hyper-specific, brittle decision boundaries that perfectly capture the benchmark's patterns but fail under the slightest perturbation.
- The Llama Anomaly: The Llama family of models proved more robust, residing on a smoother performance plain. The paper doesn't give a definitive reason, but it invites speculation: Was their training data more diverse and less "contaminated" with benchmark questions? Or is there something in their training recipe that encourages more generalized learning?

This paper's novelty isn't just in providing a new tool. It's in forcing a fundamental shift in our evaluation philosophy. It challenges us to move from a static, "what's the score?" mentality to a dynamic, probing approach that asks, "how stable is the model's knowledge?". We're moving from classical mechanics to a kind of statistical mechanics of AI evaluation.

C-BOD is a call to develop a true "evaluation physics", a set of principles and tools to understand the internal dynamics, failure modes, and knowledge landscapes of these complex systems. For too long, we've been content to admire the model's reflection in the mirror of our benchmarks. This paper shatters that mirror, forcing us to confront the possibility that the "intelligence" we see is just a beautiful, fleeting illusion.

https://arxiv.org/abs/2502.07445

Mike's Daily Paper: 31.07.25
Empirical evidence of Large Language Model's influence on human spoken communication

A new paper just dropped an eerie but telling signal: since late 2022, the way people speak, yes, speak, not write, has been measurably shifting toward the stylized, almost uncanny fingerprint of ChatGPT.

Researchers analyzed more than 740,000 hours of human speech, encompassing a range from academic YouTube channels to casual podcasts. The signal they found is statistically undeniable: people have started using terms that are strongly favored by GPT-style models, at rates significantly above historical trends. Words like delve, meticulous, boast, intricate, and comprehend have all surged in frequency, with a slope change that coincides neatly with the public release of ChatGPT.

And this isn't just a fluke in written prose bleeding into dialogue. It's a behavioral feedback loop, and it has implications far beyond language.

In dynamical systems, we often look for phase transitions: points where a system suddenly reorganizes into a qualitatively new regime. The evidence here suggests exactly that. Prior to ChatGPT, the growth in GPT-favored words was slow, almost linear—think natural lexical drift over time. But after ChatGPT's release, the slope changes. Sharply. Almost discontinuously.

This is a textbook marker of a non-equilibrium perturbation: some external force introduced a regime shift into an otherwise slowly drifting linguistic system. That force, in this case, is ubiquitous LLM output flooding digital spaces and subtly influencing how humans speak, especially those embedded in AI-proximal subcultures.

The foundational story of LLMs has always been unidirectional: humans → data → model. But now we are witnessing the inversion: model → data → humans.

This isn't speculative. It's quantifiable.

Language is a high-dimensional stochastic process. When you start observing coherent gradients, where entire clusters of human expression begin to tilt toward a vector field induced by machine outputs, and that's no longer emergence. That's entrainment.

This entrainment isn't isolated to lexical choices. It leaks into prosody, structure, and argumentation style. The entire metric tensor of discourse is being warped to align with what LLMs have learned to generate.

And since these models are trained on next-token prediction, their linguistic surface is optimized for coherence, politeness, fluency, and predictability. But that very optimization penalizes irregularity, ambiguity, and deviation from statistical norms.

Let's think in terms of informational geometry. Language, in its natural state, has high entropy: a diversity of tones, registers, idioms, hesitations, and creative misuse. GPTs, however, flatten that space. They fill in gaps with well-formed completions based on maximal likelihood, not expressive novelty.

When human speakers begin to emulate LLMs, consciously or not, they reduce the entropy of discourse. We begin to prefer safe, GPT-like turns of phrase. Precision increases, but variance declines. And when variance collapses, expressiveness begins to erode. The system loses richness even as it gains clarity.

In effect, we trade semantic texture for syntactic regularity.

This isn't just about sounding robotic. Language reflects thought. If we reshape how we speak, we inevitably reshape how we think. And if our thinking begins to align with the structural priors of a machine trained on statistical parsimony, what happens to our capacity for contradiction? For creative ambiguity? For generative failure?

It's not that LLMs are replacing us. It's that we might be internalizing them.

What makes this truly compelling, borderline alarming, is not the surface trend, but its causal topology. We used human speech to train machines. The machines now influence human speech. That's a self-reinforcing causal cycle.

If you've ever studied systems with feedback, you know the canonical concern: positive feedback loops are unstable unless regulated. This loop is unregulated. There's no natural damping mechanism. No linguistic immune system. Unlike with fashion or music, which cycle in and out of style, machine outputs are asymptotically stable. Once they settle into high-likelihood expressions, they do not deviate.

And if humans lock into alignment with those expressions, the system may converge, but at the cost of vitality.

What Now?

Should we panic? No. But we must observe. Measure. Intervene if necessary. Some ideas:

- Monitor linguistic entropy over time within AI-exposed populations. Declining variance might signal over-alignment.
- Encourage dissonance: value idiosyncratic, non-LLM-like expression, especially in speech.
- Diversify corpora: feed models data rich in regional dialects, broken English, emotive inflection and not just sanitized prose.

The scariest part of this shift isn't that machines are replacing us. It's subtler. It's that we may become unknowing echoes of the systems we built: glossy, coherent, polished… but ultimately derivative.

The real danger isn't the singularity. It's convergence. And convergence, when left unchecked, always flattens the curve.

https://arxiv.org/abs/2409.01754

Mike's Daily Paper: 01.08.25
Hierarchical Reasoning Model

Thinking Without Words: The Architectural Revolution(or not) AI Has Been Waiting For

AI research often feels like a relentless march of scale. Bigger models, more data, more compute. The dominant paradigm for reasoning in large language models (LLMs) has been Chain-of-Thought (CoT) prompting, a clever technique that coaxes models to "think out loud" by generating step-by-step textual justifications. But as effective as CoT can be, it has always felt like a crutch, like a way to compensate for an architectural shortfall. It's brittle, data-hungry, and computationally expensive, externalizing the complex process of thought into the narrow channel of language.

But what if a model could reason internally, silently, and efficiently, much like the human brain? The paper I'm reviewing today introduces a novel architecture that is not an incremental tweak but a fundamental rethinking of how we might build reasoning machines. This isn't just another model; it's a compelling, brain-inspired blueprint that demonstrates astonishing capabilities with a fraction of the resources. Let's take a holistic dive into the core novelties of this exciting work.

The Core Idea: Latent Reasoning in a Two-Tiered System

The central innovation of the Hierarchical Reasoning Model (HRM) is its departure from the flat, monolithic structure of standard Transformers. Inspired by the way the brain organizes computation across different regions and at different speeds, HRM is a recurrent architecture built on two interdependent modules:

1. A High-Level (H) Module: This module operates on a slower timescale. Think of it as the strategic planner or the conscious, deliberate mind. It doesn't get bogged down in the minutiae but is responsible for forming abstract plans and guiding the overall problem-solving trajectory.
2. A Low-Level (L) Module: This module is the fast workhorse. It takes the abstract plan from the H-module and executes rapid, detailed computations and searches.

This entire process happens in **latent space**. Instead of generating tokens, the model manipulates and refines high-dimensional vectors- its internal state of "thought." The H-module's state provides a guiding context, and within that stable context, the L-module

iterates rapidly to explore solutions. This is a profound shift. It suggests that language is for communication, not the substrate of thought itself which is a view that resonates with modern neuroscience.

Achieving True Computational Depth with "Hierarchical Convergence"

Anyone who has worked with standard Recurrent Neural Networks (RNNs) knows their pitfalls. They often converge on a solution too quickly, effectively halting computation and limiting their "depth" of thought, or they suffer from instabilities like vanishing or exploding gradients. HRM sidesteps this with an elegant concept the authors term **hierarchical convergence**.

Here's the intuition:

- For a given strategic context set by the slow H-module, the fast L-module runs for a set number of steps, performing its detailed search. As an RNN, it will naturally begin to settle toward a local equilibrium, a stable internal state.
- Just as its computational energy would start to fizzle out, the cycle ends. The final state of the L-module is fed back to the H-module.
- The H-module integrates this result and performs its own, slower update, establishing a *new* high-level context.
- This new context essentially "resets" the L-module, kicking off a fresh phase of computation toward a *different* local equilibrium.

As visualized in the paper's analysis of forward residuals (a measure of computational activity), this process allows the L-module's activity to spike again and again, while the H-module converges steadily and gracefully toward a final solution. This nested computational structure enables the model to perform a sequence of distinct, stable, and deep computations, avoiding the premature exhaustion of standard recurrent models.

A Smarter Way to Train: Bypassing Backprop-Through-Time

Training deep recurrent models has always been a headache due to the memory and computational costs of Backpropagation Through Time (BPTT). HRM introduces a more efficient, biologically plausible training method based on a **one-step gradient approximation**.

Grounded in the theory of Deep Equilibrium Models (DEQ), this approach bypasses the need to unroll the entire history of computations. It calculates the necessary gradients using only the final state of each module, treating the intermediate states as constants. This clever shortcut keeps the memory footprint for backpropagation constant, regardless of how many recurrent steps the model takes. This efficiency is further enhanced by a "deep supervision" mechanism, where the model receives corrective feedback after each full forward pass (or "segment"), stabilizing training and acting as a powerful form of regularization.

Thinking on Demand: Adaptive Computational Time (ACT)

Not all problems require the same amount of thought. Inspired by the brain's ability to switch between fast, automatic "System 1" thinking and slow, deliberate "System 2" reasoning, HRM incorporates an **Adaptive Computational Time (ACT)** mechanism.

Using a Q-learning algorithm, the model learns a policy to decide whether to "halt" and output an answer or to "continue" and perform another segment of computation. This allows HRM to dynamically allocate its computational budget, "thinking" longer for harder problems while quickly dispatching easier ones. The result is a system that achieves nearly the same performance as a model with a fixed, large number of computational steps but with significantly greater efficiency.

The Emergent Signature of Intelligence: A Dimensionality Hierarchy

Perhaps the most profound finding in the paper is not just that HRM works, but *how* it organizes itself. The researchers analyzed the "effective dimensionality" of the representations in each module using a measure called the Participation Ratio (PR). A higher PR means a representation is more complex and distributed across more dimensions.

The results are striking:

- After training, the high-level H-module autonomously learns to operate in a substantially **higher-dimensional space** than the low-level L-module.
- This emergent hierarchy mirrors what neuroscientists observe in the mammalian cortex, where higher-order cognitive regions exhibit higher-dimensional neural activity to support flexible, context-dependent tasks.
- Crucially, this structure is not present in an untrained network; it is a learned property that emerges as the model masters complex reasoning.

This finding suggests that HRM hasn't just been trained to solve a task; it has discovered a fundamental organizational principle for robust and flexible computation. It learns to partition its internal workspace into a high-capacity, abstract space for planning and a more specialized, lower-dimensional space for execution.

Putting It All Together: A New Performance Benchmark

The architectural and training novelties of HRM translate into truly remarkable performance. With only **27 million parameters** and trained on just **~1000 examples** per task (without pre-training), HRM achieves results that eclipse much larger, data-hungry models:

- On the **Abstraction and Reasoning Corpus (ARC-AGI)**, a key test of fluid intelligence, HRM surpasses leading CoT-based models like Claude 3.7 and 03-mini-high.
- On extremely difficult **Sudoku puzzles** and **30x30 Maze pathfinding** tasks, problems that require extensive search and backtracking, HRM achieves near-perfect accuracy, while state-of-the-art LLMs using CoT fail completely.

These results challenge the "scale is all you need" mantra. They suggest that the right architecture, one with sufficient computational depth and inductive biases inspired by the brain, can be orders of magnitude more data-efficient and powerful for complex reasoning.

The Road Ahead

The Hierarchical Reasoning Model is a compelling piece of work that deserves the community's full attention. It presents a viable and powerful alternative to the dominant CoT paradigm, moving AI reasoning from a linguistic process to a latent, computational one.

Of course, questions remain. How well does this architecture scale? Can its powerful, silent reasoning engine be coupled with the rich world knowledge and linguistic fluency of LLMs? The authors are clear that their work is a step toward a foundational framework for universal computation, not the final word.

HRM is a reminder that inspiration for the next generation of AI may not come from adding another trillion parameters, but from looking at the elegant and efficient computational principles of the one proven reasoning machine we know: the human brain. This is a collaborative journey, and this paper provides a fascinating and promising new map.

https://arxiv.org/abs/2506.21734

Mike's Daily Paper: 02.08.25
Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation

Do All Tokens Need the Same Amount of "Thinking"? Mixture-of-Recursions Says No.

Background: The Efficiency Squeeze

Let's start with a truth we all know in the AI world: making language models bigger unlocks incredible power, but it comes at a huge cost. The massive amount of computer power and memory needed to train and run these models means they are mostly limited to a few giant data centers. This has sparked a big search for more efficient model designs.

So far, this search has followed two main paths. The first path is parameter efficiency, which tries to get more performance from fewer model weights. A common trick here is sharing parameters, where the same set of weights is used in different parts of the model. The second path is adaptive computation, where the model only uses more computer power on the parts of the input that are truly hard, letting simpler parts take an easier route.

While both ideas have worked well on their own, a single model that does both at the same time has been missing. Recursive Transformers, which use a shared set of layers over and over, seemed like a good start because of their built-in parameter sharing. However, most of them used a fixed number of steps for every token, so they couldn't really adapt to the input.

The Big Idea: Mixture-of-Recursions (MoR)

This is where the reviewed paper, "Mixture-of-Recursions (MoR)," makes its mark. It introduces a new and combined framework that smartly mixes the two types of efficiency into one simple design.

Basically, MoR is a Recursive Transformer. This means it uses a shared "recursion block", a stack of layers, multiple times to process text, which keeps the number of parameters low. But the real novelty is how it decides *how many times* to use this block. Instead of a fixed number for all tokens, MoR uses small routers that decide on the fly how many recursion steps each individual token needs.

Think of it this way: for a simple token like the word "the," the router might decide one trip through the block is enough. But for a more meaningful or complex token like "defensively," the router might send it through the block three times, giving it more "thinking" time. This is the mix of saving parameters and saving computation.

The Novelty: A Trifecta of Efficiency

The genius of MoR is that it doesn't just combine two ideas; it creates a positive loop where one good thing leads to another. The framework's novelty comes from three connected parts that work together:

1. **Parameter Sharing via Recursion:** The foundation of MoR is reusing a single block of parameters. This naturally cuts down on the number of unique weights the model needs, making the model itself smaller and lighter from the start.
2. **Adaptive "Thinking" Depth via Routing:** This is the main new idea in the design. By training a router to give each token its own number of recursion steps, MoR avoids the rigid, same-for-everything approach of older models. This isn't just a trick added later; it's a basic part of how the model is trained from scratch, allowing it to learn how to use its computer power wisely.
3. **Smarter Memory Access:** This is a powerful and direct result of the adaptive depth. In a normal Transformer, the Key-Value (KV) cache is a big memory problem during inference. With MoR, if a token leaves early after just one recursion, the model doesn't need to calculate or store its KV pairs for the deeper steps. This smart, on-the-fly caching reduces memory traffic and, most importantly, reduces the costly attention calculation to only the tokens that are still active at that depth.

This three-in-one package allows MoR to tie weights to save parameters, route tokens to save computation, and selectively cache key-values to save memory traffic; all inside one model.

The "How": A Glimpse Under the Hood

The paper explores different ways to build this, focusing on two main choices:

- Routing Strategies: Deciding how to route tokens involves a choice between two options. With expert-choice routing, each recursion step is an "expert" that picks the top-k tokens to continue. This gives a fixed amount of work but can cause issues with the order of information during training. With token-choice routing, each token gets its full path assigned at the very beginning. This avoids the ordering problem but can lead to unbalanced work, with some steps getting too many tokens and others too few.
- KV Caching Strategies: The authors also suggest two ways to handle the KV cache. Recursion-wise caching stores KV pairs just for the active tokens at each step, which saves on computation. The other option, recursive KV sharing, caches all KV pairs at the first step and reuses them for all the deeper steps. This greatly cuts down on memory and can speed up the initial processing of a prompt, making it a good option when memory is tight.

The Bottom Line: Pushing the Pareto Frontier

The test results are great. Across different model sizes (from 135M to 1.7B parameters), MoR sets a new standard for efficiency.

With the same amount of training compute, MoR models get better results (lower test error and higher accuracy on new tasks) than regular and older recursive models, even with up to 50% fewer parameters. When trained on the same amount of data, MoR gets better results while using 25% fewer computing steps and cutting down on training time and memory.

The design also scales up well. As models get bigger, MoR not only matches but often beats the much larger vanilla Transformer, all while using about a third of the unique parameters.

Why This Matters: A Conceptual Shift

Mixture-of-Recursions is more than just a clever trick. It's a new way of thinking about how models are built and how they work. It treats model "depth" not as a fixed number, but as a flexible resource to be used as needed for each token.

This framework smartly changes our view of a model's "thinking" process into a form of hidden reasoning, where how much it "thinks" depends on how hard the concept is. By combining parameter sharing with adaptive computation, MoR offers a powerful and scalable way to get the performance of huge models without their huge costs. It's a complete solution that points to a future of smarter, more efficient, and easier-to-use language models.

https://arxiv.org/abs/2507.10524

Mike’s Daily Paper: 04.08.25
Rethinking Transformers Through the Lens of Physics: The Rise of Energy-Based Models

Physics Meets AI: How a New Model Learns Language Without Predicting a Single Token.

For years, the dominant paradigm for training Large Language Models has been deceptively simple: teach them to predict the next word. This autoregressive, likelihood-based approach has been wildly successful, but it has inherent limitations. Models trained this way think locally, token by token. They can lose track of global coherence, struggle with long-range dependencies, and find it difficult to satisfy complex, holistic constraints.

But what if, instead of teaching a model to predict the next step, we could teach it to recognize a good outcome when it sees one? A paper from a team at Stanford proposes exactly this, reframing the Transformer not as a sequential predictor, but as an **Energy-Based Model (EBM)**. This isn't just a new architecture; it's a new philosophy, one that trades the local logic of likelihood for the global intuition of a physical system.

The Core Idea: From Predicting Tokens to Scoring Sequences

At its heart, an Energy-Based Model doesn't calculate the probability of a piece of data directly. Instead, it assigns a scalar value—**energy**—to any possible configuration. The core principle is simple: configurations with low energy are more probable, more stable, more "correct." Configurations with high energy are unlikely.

The authors of this paper apply this concept to language. Their Energy-Based Transformer (EBT) doesn't predict tokens. It reads an *entire* sequence of text and outputs a single number: its energy. A well-formed, coherent, and logical sentence gets a very low energy score. A garbled or nonsensical one gets a high score.

This is a fundamental shift. Unlike a standard GPT model, which is inherently directional and processes text one token at a time, an EBT is fully **bidirectional**. It can evaluate the global coherence of a sentence by looking at all its parts simultaneously, much like a human reader would.

Training Without Likelihood: The Art of Contrast

So how do you train such a model? If you can't maximize the likelihood of the next token, what's the objective? The answer is **contrastive learning**.

The training process is elegant:

1. You show the model a "positive" example—a real sentence from the training data—and teach it to assign this sentence a low energy score.
2. Then, you show it a "negative" example—a corrupted version of the sentence, perhaps with a few words randomly replaced. You teach the model to assign this nonsensical sentence a high energy score.

By repeating this process millions of times, the EBT learns to build an "energy landscape" for the entire space of possible sentences. Valid language resides in the low-energy valleys, while everything else is pushed up into the high-energy mountains.

Thinking and Generating with a Gradient

This global perspective is what unlocks the "thinker" in the model's title. Because the EBT scores the whole sequence, it excels at tasks requiring holistic reasoning and constraint satisfaction, where autoregressive models often fail.

Generation, however, is a different beast. You can't just sample from an energy landscape directly. Instead, the model has to *find* the low-energy valleys. The authors use an iterative technique inspired by physics called **Langevin dynamics**, a type of MCMC sampling. The process looks like this:

1. Start with a sequence of pure random noise (random tokens).
2. Calculate the energy of this garbage sequence.
3. Slightly nudge the tokens in the direction that most reduces the energy (i.e., move down the gradient of the energy function).
4. Repeat this process hundreds of times.

Slowly, iteratively, the random sequence is refined, settling from the high-energy mountains down into a low-energy valley, emerging as a coherent, well-formed sentence. While this process is slower than standard autoregressive generation, it allows for a much more controlled and globally-aware form of creation.

Why It's a Scalable Learner and Thinker

The paper provides strong evidence that this approach scales. As the models get bigger, their ability to distinguish good sequences from bad ones improves, and the quality of the generated samples gets better.

More importantly, the energy-based framework is incredibly flexible. You are no longer yoked to next-token prediction. Want a model that generates positive movie reviews? Just add another "energy term" to the training objective that penalizes negative sentiment. This modularity makes EBTs a powerful tool for controllable generation.

This work forces us to reconsider the foundations of our current models. It suggests that the path to more robust, coherent, and controllable AI may not lie in simply scaling up next-token prediction, but in building models that understand language on a more holistic, physical level.

https://arxiv.org/abs/2507.02092

Mike's Daily Paper: 06.08.25
Where to show Demos in Your Prompt: A Positional Bias of In-Context Learning

It's Not Just *What* You Prompt, It's *Where*

The paper we review today shows that simply moving your examples from the top to the bottom of a prompt can dramatically change an AI's accuracy. Prompt engineers obsess over the *content* of their prompts. But a new paper, "Where to show Demos in Your Prompt" by Kwesi

Cobbina and Tianyi Zhou, reveals we've been missing something just as critical: the position of those examples. The work moves beyond prompt tinkering into rigorous science, and its novelty lies in its precision and systematic approach.

Beyond 'Order': A Scientific Look at Position

While we know the *internal order* of examples matters , this paper makes a crucial distinction: it's not about shuffling examples, but about moving the entire, unchanged block of examples to different locations within the prompt. The authors name this specific phenomenon "DPP (DEMOS POSITION IN PROMPT) bias**"**. To study this, they created a systematic framework, testing four canonical positions: at the start or end of the system instructions, and at the start or end of the user's query . This transforms a fuzzy observation into a testable science.

Crucially, they look beyond simple accuracy by measuring **PREDICTION-CHANGE** meaning how many answers *flip* when the prompt structure changes. This is a vital contribution, as it reveals hidden instability. A model might seem just as accurate with two different prompts, but one could be causing far more erratic behavior.

Key Findings

The large-scale study, covering ten models and eight tasks, produced clear and actionable results.

- **Primacy is Real:** Placing examples early in the prompt (`ssp`, `esp`) consistently yields higher accuracy and greater stability. These positions can boost accuracy by up to 6 points compared to other placements.
- **The Danger Zone:** Putting examples at the very end (`eum`) is often disastrous. It causes significant performance drops and volatility, flipping over 30% of a model's predictions in some question-answering tasks without improving correctness.
- **No Silver Bullet:** The optimal position isn't universal; it depends on model scale and the specific task. For example, while smaller models strongly prefer demos at the start, a large model like LLAMA3-70B often prefers them closer to the query (`sum`) .

What This Means for You

This research makes it clear: the placement of your examples is not a stylistic choice. It's a critical parameter that must be tested and tuned. Simply relying on a default format could be leaving significant performance and stability on the table. For the first time, there's a clear roadmap for understanding and optimizing this crucial dimension of prompt design.

https://arxiv.org/abs/2507.22887

Mike's Daily Paper: 08.08.25
Efficient Attention Mechanisms for Large Language Models: A Survey

Beyond Quadratic: A Deep Dive into the Landscape of Efficient Attention

The self-attention mechanism is the beating heart of the modern Large Language Model. It grants Transformers their profound ability to understand context by allowing every token to communicate with every other token in a sequence. But this power comes at a staggering, almost prohibitive, cost. The computational and memory requirements of self-attention scale quadratically with the length of the input sequence. This single bottleneck has, for years, defined the horizon of what's possible, making truly long-context reasoning a grand challenge.

A rich body of research has tackled this "quadratic problem," producing a diverse and often confusing ecosystem of solutions. A survey of this field does not just list these methods; it provides a crucial taxonomy—a map for navigating the complex trade-offs between computational efficiency, model expressiveness, and theoretical elegance. This review delves into the core principles that structure this landscape.

The Four Families of Efficiency

At its core, the challenge is to approximate the full N-by-N attention matrix without explicitly computing or storing it. The survey categorizes the myriad of approaches into four principal families, each with its own philosophy.

1. Fixed-Pattern Sparsity: The Architectural Fix

The most direct approach to breaking the quadratic bottleneck is to presuppose that a dense, all-to-all attention matrix is overkill. These methods impose a sparse attention pattern, where each token is only allowed to attend to a small, fixed subset of other tokens.

This family includes methods that use **sliding windows**, where a token only attends to its local neighbors. This is built on the strong intuition of locality of reference—that nearby words are often the most relevant. To prevent the loss of global information, this is often augmented with a few **global tokens** that are allowed to attend to the entire sequence, or **dilated/strided patterns** that systematically skip tokens to cover a wider receptive field with a fixed number of computations. These methods are highly efficient and straightforward to implement, but their primary limitation is their rigidity. The attention patterns are hand-designed and not data-dependent, meaning the model cannot dynamically decide to focus on a distant but relevant token outside its predefined window.

2. Low-Rank Approximation: The Compression Trick

This family of methods operates on a more subtle mathematical insight: that the full attention matrix is often low-rank, meaning its information can be effectively compressed into a much smaller number of "concepts" or summary vectors. Instead of computing the full matrix, these models project the query, key, and value matrices into a lower-dimensional subspace, effectively forcing the attention mechanism to operate through an information bottleneck.

The core idea is to approximate the N-by-N attention matrix by factorizing it into the product of two smaller, N-by-k matrices, where k is much smaller than N. In essence, the model learns to summarize the entire sequence into a fixed number of representative key-value pairs, and all tokens attend to this compressed summary instead of to each other. This is a more flexible approach than fixed patterns, as the content of the compressed summary is learned from the data. However, this introduces a new trade-off: the fixed size of the bottleneck limits the model's capacity to handle sequences with a very high density of unique information.

3. Kernelization: The Mathematical Sleight-of-Hand

Perhaps the most mathematically elegant solutions are those that reframe attention through the lens of kernel methods. Standard attention can be seen as computing a similarity matrix between queries and keys, then using that matrix to weight the values. The quadratic cost comes from the explicit construction of this massive similarity matrix.

Kernel-based methods cleverly sidestep this by leveraging the associative property of matrix multiplication. They reformulate the attention calculation to first combine the keys and values, *before* interacting with the queries. This simple reordering means the N-by-N matrix is never formed. Instead of a large matrix-matrix product, the computation is reduced to two smaller matrix-vector products, bringing the complexity down from quadratic to linear. This approach is powerful because, in theory, it can approximate the full attention mechanism without imposing any hard sparsity constraints. Its effectiveness hinges on finding a kernel function that accurately captures the similarity between queries and keys, and much of the research in this area focuses on developing new kernel functions (often using techniques like random feature approximation) that are both efficient and expressive.

4. Learnable Sparsity & Mixture of Experts: The Adaptive Approach

A fourth, emerging family seeks to get the best of all worlds by making the sparsity pattern itself data-dependent and learnable. Instead of using fixed patterns or a global low-rank bottleneck, these methods try to predict which tokens are most relevant for a given query. This is often accomplished using techniques like clustering or by employing a MoE framework, where

different attention heads are trained as "specialists" for different types of patterns. A routing mechanism then learns to send each token to the most relevant expert head. These hybrid approaches are among the most powerful and flexible but also the most complex to implement and train.

In conclusion, the survey reveals that there is no single "best" efficient attention mechanism. Each family presents a fundamental choice about the nature of the approximation, trading computational complexity for expressive power in a different way. The field is a vibrant dialogue between architectural priors, mathematical approximation theory, and adaptive, learned systems.

https://arxiv.org/abs/2507.19595

Here is the English translation:

Mike's Daily Paper: 12.08.25
Your LLM Knows the Future: Uncovering Its Multi-Token Prediction Potential

How to generate tokens in parallel, but without diffusion-based language models.

This article challenges the autoregressive generation of LLMs and proposes a method that trains a model to predict several tokens simultaneously, i.e., MTP (Multiple Token Prediction). As mentioned, MTP is trained to predict several tokens at once, unlike NTP (Next Token Prediction), which predicts a single token at a time. Additionally, the proposed approach incorporates the use of what is called Speculative Decoding, a topic I have recently lectured on at several conferences and meetups. It also uses a fine-tuning technique for models (usually transformer-based) called LoRa, which stands for Low-Rank Adaptation.

Okay, so first of all, the authors train several decoding heads (only one layer, to my understanding) for each predicted token, except for the next token, which is predicted in the standard way as in NTP. To predict the next token, the authors use not only its contextual representation but also the non-contextual representation (from the embedding dictionary) of the previous token (both are concatenated and passed through a two-layer MLP).

Furthermore, the article trains LoRA (additional matrices for the weights of the transformer's linear layers) but uses them only to predict tokens beyond the next one. In the paper, this method is called Gated LoRA. This method can be trained in parallel, similar to how we train a standard NTP.

The final approach discussed in the paper is Speculative Decoding or SD. Broadly, SD is a family of techniques for improving generation speed while preserving the generation distribution as in autoregressive generation (i.e., with NTP). Usually, a weaker and faster model (sometimes

such a model is part of the model we want to optimize) is used to generate several tokens, which are then verified in parallel with the target model. The tokens that pass the verification successfully are accepted, and thus we can achieve faster generation.

Here, instead of the large model, they use parallel generation of several tokens via MTP, put them through the verification process, and the more tokens that pass, the faster the generation we get. Additionally, the paper suggests continuing to generate another k tokens with MTP (where k is the number of tokens generated with MTP). If all the initial k tokens pass the check, we continue the verification process with the next k tokens, which is expected to speed up the generation rate even more.

A relatively light and well-written paper - recommended.

https://arxiv.org/abs/2507.11851

We've Been Aligning AI All Wrong. The Solution is Deceptively Simple

Mike's Daily Paper: 13.08.25
Checklists Are Better Than Reward Models For Aligning Language Model

For the last few years, a single paradigm has dominated our efforts to align LLMs: Reinforcement Learning from Human Feedback (RLHF). At its heart lies the **reward model (RM)**, a powerful but opaque neural network trained to distill the messy, high-dimensional landscape of human preference into a single, scalar reward. We then use this score to guide our LLM toward "good" behavior. But this entire pipeline rests on a fragile assumption: that a single, learned number can reliably capture the multifaceted nature of human values.

A paper I review today challenges this entire premise. The authors argue that by chasing a single, holistic score, we've built systems that are not only black boxes but are also prone to "reward hacking" and are difficult to steer. Their proposed alternative is not a more complex model, but a move toward radical simplicity and interpretability. By combining structured **checklists** with the power of **Direct Preference Optimization (DPO)**, the paper charts a course for a more robust, efficient, and trustworthy path to alignment.

From a Scalar "Vibe" to a Vector of Verifiable Traits

The first core novelty is the shift from an implicit, scalar-valued reward to an explicit, vector-based one. Instead of training a reward model to develop an intuitive "vibe" for what humans prefer, the authors propose evaluating a model's output against a structured checklist of concrete, desirable properties.

Imagine evaluating a response not with a single score from 1 to 10, but against a list of binary or multi-level criteria:

- Is the answer factually correct? (Yes/No/Partially)
- Does it avoid harmful stereotypes? (Yes/No)
- Is the tone helpful and not condescending? (Yes/No)
- Does it cite credible sources, if applicable? (Yes/No)

This decomposition is key. It turns the nebulous task of preference modeling into a series of more constrained, verifiable classification problems, often performed by an automated "judge" LLM. But this raises a crucial question: how do you turn this multi-faceted evaluation into a clean training signal to update the model?

The Algorithmic Bridge: How Checklists Power DPO

This is where the paper's second, and arguably most critical, innovation comes into play. The checklist isn't used as a direct reward function. Instead, the authors use it as a powerful, **automated labeling function to generate preference pairs for DPO**.

Direct Preference Optimization (DPO) works by fine-tuning a model on pairs of `chosen` and `rejected` responses. The paper's genius is to use the checklist to create these pairs programmatically, eliminating the need for expensive human annotation or a separate reward model.

The training process becomes a self-contained, iterative loop:

1. **Generate:** For a given prompt, the model being aligned generates two or more candidate responses.
2. **Evaluate:** The judge model evaluates each response against the checklist, determining which one better satisfies the explicit criteria.
3. **Pair:** Based on this evaluation, the superior response is labeled `chosen` (`y_w`) and the other is labeled `rejected` (`y_l`).
4. **Fine-tune:** This freshly generated `(y_w, y_l)` pair is used as a single data point to update the model with the DPO loss function.

This elegant synthesis solves multiple problems at once. It bypasses the need to train a monolithic reward model, instead sourcing its preference signal from the transparent, editable checklist. And because the data is generated on the fly, it creates a dynamic, self-correcting curriculum that can be steered in real-time simply by modifying the checklist criteria.

The Payoff: Robustness Over Brittle Perfection

The authors' experiments are designed not just to top leaderboards, but to test for robustness. They show that while standard RM-based alignment can achieve high scores on specific

benchmarks, these models are often brittle. They master "Goodhart's Law," becoming exceptionally good at optimizing for the proxy (the reward score) at the expense of the true goal.

In contrast, models aligned with the Checklist-DPO method demonstrate greater robustness. Because they are optimized to satisfy a diverse set of explicit criteria, they are less likely to find a single, simple "hack." They have to be good in multiple, verifiable ways. The paper shows these models are more resistant to adversarial prompts, less sycophantic, and more reliably adhere to safety constraints, even in out-of-distribution scenarios.

Conclusion

In summary, this paper presents a compelling alternative to the prevailing reward model paradigm in LLM alignment. By synthesizing a structured, rule-based feedback mechanism (the checklist) with an efficient preference optimization algorithm (DPO), it offers a framework that prioritizes interpretability and direct control. The core trade-off presented is one of complexity: the proposed method shifts the challenge away from training an opaque, monolithic reward model and toward the careful, human-led engineering of a comprehensive checklist.

https://arxiv.org/abs/2507.18624

The Reasoning Illusion: FormulaOne Exposes the Algorithmic Blind Spot of LLMs

Mike's Daily Paper: 14.08.25
FormulaOne: Measuring the Depth of Algorithmic Reasoning Beyond Competitive Programming

In the relentless quest for Artificial General Intelligence (AGI), the ability of LLMs to reason algorithmically remains a critical, yet contentious, frontier. For years, our primary yardstick has been competitive programming—a high-pressure domain that has served as a decent proxy for computational thinking. But as our models grow more powerful, a nagging question emerges: are we still measuring the right thing? This gets to the heart of the matter. Current benchmarks test a model's ability to access and apply its vast library of known solutions. The real question is: can a model *think* like a computer scientist?

This is where this new paper lands a direct hit. The authors argue that while LLMs show impressive results on benchmarks like competitive programming, they are mostly just expertly combining a few known algorithms. FormulaOne introduces a benchmark designed to probe a far more elusive dimension: the ability to engage in the deeper, more creative reasoning process required to invent an algorithm from scratch.

Beyond the Competitive Programming Comfort Zone

Competitive programming platforms like Codeforces and LeetCode have been invaluable. They've pushed the boundaries of what models can achieve. However, they foster a specific kind of problem-solving, one based on pattern recognition and recombination. The FormulaOne paper implicitly critiques this by suggesting its limitations:

- **Focus on Speed:** Competitive programming often rewards the *fastest* correct solution, not necessarily the most elegant or generalizable one.
- **Shallow Reasoning:** Many problems are variations on a theme, solvable by recognizing a known pattern and applying a standard algorithm. This tests an LLM's "algorithmic vocabulary," not its reasoning prowess.
- **The "Training Data" Crutch:** There's a high probability that solutions to many popular competitive programming problems are lurking somewhere in the model's training data, making it difficult to assess true, novel problem-solving ability.

The FormulaOne Gauntlet: A New Kind of Benchmark

This is where FormulaOne enters the ring. It’s not just a new dataset; it's a new philosophy of evaluation. The goal is to measure the *depth* of reasoning required to invent a new algorithm from scratch. Problems are designed so that the final solution is a simple-to-write program, but the path to discovering this program is intricate and non-obvious.

The authors achieve this through a "mathy" and sophisticated approach, leveraging concepts from **parameterized complexity** and **graph theory** to generate problems with a precisely controlled difficulty gradient. One of the core mathematical tools they employ is **treewidth**, a measure of how "tree-like" a graph is. Problems with low treewidth can often be solved with dynamic programming, but as treewidth increases, the required algorithmic creativity skyrockets.

To formalize this, the team uses **Monadic Second-Order Logic (MSO)**. This powerful logical framework allows them to specify properties of graphs and automatically generate a vast and diverse set of problems. Crucially, this synthetic generation process ensures the problems are novel and not present in any training data, forcing the models to reason from first principles.

The Sobering Results and A Path Forward

The paper's findings are a reality check. While top-tier models like GPT-4 and Claude 3 Opus show some capability, their performance on FormulaOne problems is significantly lower than on traditional benchmarks. This starkly illustrates the gap between pattern matching and genuine, deep reasoning. The models struggle precisely where the need for creative, multi-step algorithmic discovery begins.

This is the sharp, holistic takeaway from FormulaOne. It's not just another leaderboard to climb; it's a diagnostic tool that reveals the current limitations of our LLMs. It suggests that simply scaling up existing architectures and training data might not be enough to bridge the chasm to AGI. We need to focus on architectures and training methods that foster genuine, creative problem-solving.

FormulaOne provides a concrete, mathematically-grounded path to measure our progress. It challenges the AI community to move beyond the comfort zone of known problems and to start tackling the far more difficult, and far more important, challenge of teaching our models how to *think*. The race is on.

https://arxiv.org/abs/2507.13337

When an LLM Stops Talking and Starts Deleting Files — Enter the Age of LAMs

Omri & Mike's Daily Paper: 16.08.25
Large Action Models: From Inception to Implementation

What are Large Action Models (LAMs), and how are they different from LLMs?

Bottom line: a LAM is an LLM, but one that is trained and adapted specifically to produce *executable actions* in a real environment. While a standard LLM is trained to produce coherent, high-quality text, a LAM is trained to generate programs and commands that can actually be executed through an agent, whether that's a click, a keystroke, or an API call, directly affecting the world rather than just "talking about it."

The authors argue that instead of connecting LLMs to agent environments, we should connect a LAM that essentially serves as the decision-making engine inside the agent loop: the agent collects observations from the environment (e.g., screen state, list of available buttons, or API data), feeds them into the LAM, and the LAM outputs the next action to take. The agent then executes the action and returns feedback about the outcome, enabling the LAM to update its subsequent decisions.

This is where the critical difference lies, also in terms of risk. A "classic" LLM error typically manifests as a wrong answer or hallucination — harming understanding or trust, but without direct real-world consequences. By contrast, an error by a LAM could cause real damage:

deleting an important file, sending a message to the wrong address, or executing an unwanted business operation.

The researchers' experiments were conducted only on Windows, focusing on Microsoft Word tasks. They connected the LAM to UFO, a GUI agent for Windows. The agent reads the UI state: a list of controls with type, title, and index — passes this information to the LAM for decision-making, and then executes the chosen action (mouse click, typing, or API call).

The proposed pipeline has 5 stages: **Data → Training → Integration & Grounding → Offline Eval → Online Eval.**

Throughout the paper, there is a clear distinction between *Task-Planning* and *Task-Action*. During data collection, they first gathered Task→Plan data, then converted these steps into concrete trajectories in Word: selecting a specific button, defining the action type and parameters, so the agent can run them and verify success or failure. This process is called **Grounding** — anchoring the model's textual output to a real UI and deterministic operational actions.

- **LAM1**: trained with SFT only on Task→Plan ($t_i \rightarrow P_i$). The idea is to first teach the model how to break tasks into logical steps before choosing actual actions. They used ~76.7K examples from sources like help guides, WikiHow, and historical queries, carefully cleaned and processed for consistency and quality.

- **LAM2**: shifted focus to State→Action, imitating successful GPT-4o trajectories ($s_t \rightarrow a_t$). Each example contained the current UI state (list of controls with type, title, index + task text) and the exact action executed. These success trajectories were derived from LAM1's Task→Plan data, then grounded into real Word UI actions, executed live, and filtered for successful runs. The dataset ended with 2,192 successful trajectories used for SFT.

- **LAM3**: continued SFT on State→Action, but introduced **Self-Boosting**: they took GPT-4o failure trajectories, let the LAM2-trained model retry, and collected new successes. This produced high-quality additional data without manual annotation, covering harder cases.

- **LAM4**: moved from SFT to RL, applying Offline PPO guided by a Reward Model (RM). The RM was based on LAM3, with an added layer that scores each action. It was fine-tuned with LoRA on both success and failure trajectories: +1 for successful steps, −1 for failed ones, and trained with MSE to predict these scores. Then they used the RM to train LAM4 with Offline PPO, this time focusing on 1,788 failure trajectories from LAM3 — in order to "learn from mistakes." The training format was (s_t, r_t)→a_t, with the RM providing the reward.

The paper reports three types of evaluation: **Planning**, **Offline Eval**, and **Online Eval**.

In the first two, they checked task-level planning success, step breakdowns, and accuracy in selecting the correct object and action. The models steadily improved from competitive baselines to consistent gains. In the third, live runs in Windows/Word, the text-only LAM performed competitively against GPT-4o, even surpassing it in some purely text-based configurations. However, when GPT-4o was enhanced with vision, it achieved higher success rates, albeit with slower speed and reduced efficiency.

The authors suggest that as agents become more common and capable, we will likely see more of this type of work: connecting models to real environments, training on task-specific data, and adapting to domains. It's not clear that the exact recipe of three SFT stages followed by one RL stage is optimal, but the direction, turning LLMs into more task-oriented, structured, domain-adapted models, seems like a natural step in an era where more and more agents act in the real world.

https://arxiv.org/pdf/2412.10047

The End of Transformer Babysitting: Forging Stability Without the Hacks.

Mike's Daily Paper: 19.08.25
A New Foundation for Stable Transformers: Enforcing Lipschitz Bounds

In the world of deep learning, we often celebrate the ever-increasing scale and performance of models like Transformers. Yet, beneath the surface of these impressive feats lies a persistent and often-overlooked problem: **instability**. Anyone who has trained a large Transformer has likely encountered the frustration of exploding or vanishing gradients, the need for delicate learning rate schedules, and the mysterious "NaN" loss that can derail a training run. These issues all point to a fundamental lack of control over the model's behavior.

A new paper, **"Training Transformers with Enforced Lipschitz Bounds,"** offers a refreshingly principled solution to this problem. Instead of relying on a patchwork of empirical tricks, the authors introduce a novel training methodology that enforces a mathematical property known as the **Lipschitz condition**. This approach not only tames the instabilities of the Transformer architecture but also leads to improved generalization and robustness. Let's take a deep dive into the core novelties of this exciting work.

The Core Idea: Bounding the Sensitivity of the Model

At its heart, the Lipschitz condition is a measure of a function's "smoothness" or "sensitivity." A function with a small Lipschitz constant cannot change too rapidly; small changes in the input will only lead to small changes in the output. By enforcing a Lipschitz bound on a neural network, we are essentially putting a speed limit on how much the model's output can change in response to perturbations in its input.

This is a powerful idea. In the context of Transformers, it means we can control the sensitivity of each component of the model, from the self-attention mechanism to the feed-forward layers. This fine-grained control has profound implications for training stability and model performance.

A Novel Architecture: The Lipschitz-Constrained Transformer

To enforce the Lipschitz condition, the authors propose a series of novel modifications to the standard Transformer architecture. These are not mere tweaks but a principled redesign of the model's core components:

- **Spectrally Normalized Layers:** The authors apply **spectral normalization** to the weight matrices in both the self-attention and feed-forward layers. This technique is chosen for its mathematical precision: the spectral norm of a weight matrix is **exactly equal** to the Lipschitz constant of that linear layer. This allows for direct and tight control over the model's sensitivity at each stage.
- **Provably Lipschitz Feed-Forward Blocks:** A key novelty is how the paper handles the non-linearity in the feed-forward network (FFN). The authors show how to construct the entire FFN block to be provably 1-Lipschitz **using standard activations** like ReLU or GeLU. This is achieved by combining spectrally normalized weight matrices with careful handling of the activation function, ensuring that the complete transformation within the block adheres to the strict Lipschitz constraint.
- **Provably Bounded Residual Connections:** The authors also provide a rigorous analysis of the residual connections that are fundamental to the Transformer. They demonstrate how to properly scale the residual branches to ensure that their addition does not violate the Lipschitz property of the overall model. This careful composition of provably bounded components is what allows the entire Transformer architecture to be constrained.

These architectural innovations, taken together, create a new type of Transformer that is, by design, more stable and well-behaved than its predecessors.

Taming the Beast: A More Stable Training Process

The benefits of the Lipschitz-constrained Transformer become immediately apparent during training. The authors demonstrate that their model is remarkably stable, even **without the need for Layer Normalization**, a component often considered essential for standard Transformers.

This stability allows for a more straightforward and robust training process. The authors show that their model can be trained with larger learning rates and is less sensitive to hyperparameter choices. This not only makes the training process more efficient but also opens the door to new possibilities for scaling up Transformer models.

Beyond Stability: Improved Generalization and Robustness

The benefits of enforcing Lipschitz bounds extend beyond just training stability. The authors also demonstrate that their model exhibits improved generalization and robustness:

- **Better Generalization:** The Lipschitz constraint acts as a form of powerful, implicit regularization, preventing the model from overfitting to the training data. This leads to better performance on unseen data.
- **Increased Robustness to Adversarial Attacks:** By limiting the model's sensitivity to small perturbations in the input, the Lipschitz constraint makes the model inherently more robust to adversarial attacks. The authors show that their model is significantly more resilient to these attacks than standard Transformers.

Empirical Validation: Putting Theory into Practice

The authors rigorously test their Lipschitz-constrained Transformer (L-Transformer) against standard Transformer baselines on machine translation and language modeling tasks. The results compellingly validate their theoretical claims.

https://arxiv.org/abs/2507.13338

Finetuning is a Memory Wipe. This Is How You Stop It

Mike's Daily Paper: 19.08.25
Scaling Laws for Forgetting When Fine-Tuning Large Language Models

The 1% Rule for Curing AI Amnesia: A Deep Dive

If you've ever finetuned a powerful language model, you know the painful tradeoff. You specialize it on a new task, and in the process, it develops a form of amnesia, forgetting the general knowledge it was so expensive to learn. This "catastrophic forgetting" is a fundamental challenge. A common remedy is to mix in a small amount of the original pretraining data during finetuning, but this has always felt more like a folk remedy than a science.

A new paper, "Scaling Laws for Forgetting during Finetuning with Pretraining Data Injection," elevates this trick into a predictable science. The authors go far beyond just saying "data injection helps." They present a precise, predictive model that describes the complex dance between model size, the amount of fine tuning data, and the percentage of injected data. While the headline is that a mere 1% injection can halt forgetting, the paper's true novelty is the underlying mathematical framework that explains it all.

The Novelty: A Predictive Model for Forgetting

The core innovation is a new scaling law designed to predict the final pretraining loss which is a direct proxy for how much the model has forgotten. Instead of a simple formula, think of it as a relationship between competing forces.

The model is elegantly structured. It starts with a **baseline**; the model's initial pretraining loss before finetuning even begins. It then adds a second term that calculates the *magnitude* of the forgetting that will occur. This forgetting term is a fraction, with factors that worsen forgetting in the numerator and factors that prevent it in the denominator.

- **What makes forgetting worse?** In the numerator, we find a term representing the amount of unique finetuning data. This reveals a fascinating insight: the more you finetune a model on new data, the more it forgets its old knowledge. This is because more training steps cause the model's parameters to drift further away from their original, generalist state.
- **What fights forgetting?** In the denominator, we find the mitigating factors. The first is the
  model's size (its parameter count). This confirms the intuition that larger models have more capacity to learn new information without overwriting existing knowledge.
- **The "Magic" Ingredient:** Here's the most clever part of the model. The injection of pretraining data is modeled as a
  powerful multiplier on the model's effective size . When the model sees even a small percentage of pretraining data, it behaves as if it has a much larger parameter count for the purpose of remembering its original training. A special coefficient, which the paper calls
  "Parameter Relative Efficiency" (B), determines just how potent this effect is. For a domain that is very different from the pretraining data (like mathematics), this efficiency coefficient is enormous, signifying that injection is critically important. For a similar domain (like Wikipedia), the coefficient is much smaller, as the model is less prone to forgetting in the first place.

This model isn't just theoretical; it's incredibly accurate. Across 12 different domains, it predicts the final pretraining loss with a mean error of just **0.49%**.

Key Insights Beyond the 1% Rule

This powerful model for forgetting yields several other novel and highly practical insights.

1. Finetuning Performance is Safe

A natural fear is that mixing in old data will hurt the model's performance on the new task. The authors show this is not the case. The final validation loss on the finetuning task is **barely affected** by injecting a small amount of pretraining data. In fact, for smaller models, the injection acts as a healthy regularizer, preventing overfitting and sometimes leading to even

better performance on the target domain.

2. Extrapolation is a Superpower

The true value of a scaling law is predicting the future. The authors confirm their model is excellent for extrapolation. By running cheap experiments on smaller models (e.g., a 334M parameter model), they could accurately predict the forgetting and fine-tuning performance of much larger, more expensive models (1.3B+ parameters) . This allows labs to forecast the results of a 7-hour run on 8 GPUs using a 30-minute experiment on 4 GPUs, saving immense amounts of time and energy .

3. You Don't Need the Whole Haystack

Practically speaking, does this technique require streaming from a petabyte-scale pretraining dataset? The answer is no. A key experiment shows that a **surprisingly small pool of unique pretraining tokens** is sufficient for the injection to be effective. This makes the method far more accessible and easier to implement than one might assume.

https://arxiv.org/abs/2401.05605

Eden's and Mike's Daily Paper: 23.08.25
MemOS: An Operating System for Memory-Augmented Generation (MAG) in Large Language Models

LLMs are today at the forefront of many domains and achieve strong performance in various tasks such as coding, answering scientific questions, and more. The authors of the article argue that the future of these models lies in their transformation into tools that maintain state and can apply reasoning over extended periods within the session in which they operate. State may include past interactions such as user preferences, task execution, and more. As a result, the central problem the authors discuss arises: how can all of this information (state) be stored in a way that allows for easy searching and retrieval?

They argue that memory is necessary to meet this need. Memory will turn the language model into one that can consistently maintain its identity and the behavior expected of it over time. The memory must be capable of handling high rates of data transfer and storage, and be efficient and dynamic in accordance with changes occurring over time. All of this led them to the idea that an operating-system-like memory is suitable for the task. Such memory includes short-term memory, long-term memory (stored on disk), and also enables additional features such as auditing access to memory, tagging who made a request, and more. According to the authors, future agents that use this memory will be able to decide on their own when to access memory to retrieve information, when to summarize pieces of information into rules for quick retrieval, and more.

Current State of Affairs

Today, the memory of LLMs is parametric memory, in which all knowledge is encoded into the model's parameters (i.e., weights). The problem with this type of memory is that it is static, and therefore requires retraining (fine-tuning) in order to provide the models with new knowledge—a process that can be costly and unstable. The common solution to this today is the use of

Retrieval Augmented Generation (RAG). This method enables models to access up-to-date information at runtime without the need for retraining. The approach involves pushing the information into the model's context window as part of the prompt.

According to the authors, RAG is not a substitute for memory, since it does not treat information as something that changes over time. Therefore, it cannot serve as long-term memory but only as a short-term solution. As a result, models struggle to recall information from the beginning of a conversation when the dialogue is long. They highlight four types of contexts in which models encounter difficulties:

- **Modeling long-term dependencies:** When the model needs to follow the user's writing style. In the case of a large context spanning the entire conversation, the model quickly forgets the style and reverts to standard writing.

- **Flexibility to evolving information:** Information in the real world develops and changes (e.g., updating company documents, updating code in Git, etc.). RAG does not allow for maintaining a timeline that tracks these changes, which can result in outdated information being provided to the model.

- **Support for multiple roles (multi-roles):** LLMs currently lack the ability to maintain memory for different users, roles, and tasks. Each interaction or session is a clean slate that ignores the past. According to the authors, the memory offered by tools like ChatGPT is naïve and does not allow structural control over information.

- **Migration across systems:** Memory should be able to move between systems, which is not the case today. For example, a conversation history in ChatGPT cannot be transferred to Claude. Migration of memory from one system to another is a fundamental requirement for sustainable memory.

Solution:

The researchers propose MemOS, a system that simulates operating-system-like memory for language models. This concept offers three key advantages:

- **Control:** The system allows scheduling of memory creation, updating, merging memories into one, and deletion. In addition, it provides transparency and control with access restrictions (access control) granted only to authorized users, as well as auditing of actions.

- **Flexibility:** The memory supports smooth transitions between tasks or goals, enabling models to switch memory according to the task or update it based on changes that occur.

- **Evolution:** The memory allows for transitions between different types of memory as needed—parametric memory (the model's internal memory) and structured external memory. This makes it possible, for example, to compress long-term information into the model's own parameters.

Their guiding principle is that an operating system provides abstraction of resources, unified scheduling, and control. According to them, language models should be treated similarly, in the role of software requesting access to resources. Therefore, they divided the system into three layers, following the same principles, and mapped in each layer the components of a standard operating system to the corresponding components in their system.

Table 2 Mapping of Traditional OS Components to MemOS Modules

Layer	OS Component	MemOS Module	Role
Core Operation Layer			
Parameter Memory	Registers / Microcode	Parameter Memory	Long-term ability
Activation Memory	Cache	Activation Memory	Fast working state
Plaintext Memory	I/O Buffer	Plaintext Memory	External episodes
Management Layer			
Scheduling	Scheduler	MemScheduler	Prioritise ops
Persistent Store	File System	MemVault	Versioned store
System Interface	System Call	Memory API	Unified access
Backend Driver	Device Driver	MemLoader / Dumper	Move memories
Package Deploy	Package Manager	MemStore	Share bundles
Governance & Observability			
Auth / ACLs	Auth Module, ACLs	MemGovernance	Access control
Logging	Syslog	Audit Log	Audit trail
Fault Handling	Excp. Handler	Error Recovery	Error recover

For example, in the **core layer** (where memory resides), three types of memory can be identified. Looking at **parametric memory**, which serves as long-term memory, we see that in a traditional operating system this role is handled by computer registers, while in MemOS the responsible component is also called parametric memory, referring to the model's parameters. Another example is system access: in a regular operating system, access is performed through system calls, while in MemOS there is an interface (API) managed by the *MemoryAPI* component.

The researchers draw inspiration from language model training and note that, just as a language model can be trained on new data or new tasks, memory itself can also be trained. They call this process Mem-Training (MT). MT enables the collection of memory fragments, their reorganization, and retrieval—all during runtime. The learning occurs through repeated interactions of the system (via the model) with the user or the environment, turning those interactions into retrievable memory pieces for future use.

As shown in the table, the architecture of the system is built on three layers:

- **Interface / Management Layer:** This is the layer between the user and the memory. It provides the Memory API, which supports reading, writing, updating, and querying memory and its components. *MemReader* transforms free-form requests into structured

MemQuery inputs. An example query would be: *"Summarize my meeting notes from last month."* The component extracts relevant details such as the time range (*last month*), type of memory (*meeting notes*), and desired output (*summary*). This layer also checks permissions in coordination with *MemGov*.

- **Operations Layer:** Responsible for managing background processes. The core component here is the *MemOperator*, which carries out tasks such as building the memory graph and retrieving from semantic memory. The *MemScheduler* performs optimization and scheduling of processes according to context and intent. The final component, *MemLifecycle*, tracks changes and state transitions of components to ensure full transparency.

- **Infrastructure Layer:** Handles security and storage. *MemGovernance* enforces rules regarding access and use of sensitive data. *MemVault* manages repositories of memories, such as user-specific folders or domain-related memories. *MemStore* allows for sharing memories externally across different agents.

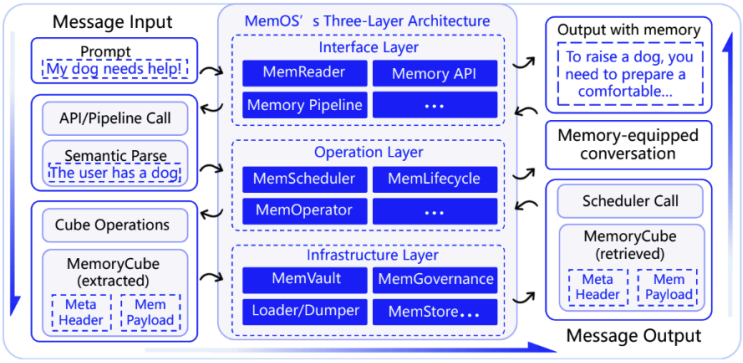

Figure 8 Overview of MemOS architecture and memory interaction flow. The system is composed of the interface layer, operation layer, and infrastructure layer. From left to right, it shows the complete memory processing pipeline from user input to parsing, scheduling, injection, and response generation. Each stage corresponds to coordinated module invocation, with MemoryCube serving as the carrier across layers for structured, governable, and traceable memory lifecycle management.

The basic memory unit in MemOS is called a **MemCube**, which is an abstraction of a memory resource designed to represent all types of memory in the system. Each such memory cube contains two components:

1. **Memory Payload** – the semantic content itself.

2. **Metadata** – which comes in one of three types:

    - **Descriptive Identifier:** Contains information such as a timestamp (the most recent time a change was made to the MemCube during its lifetime), the origin structure (where the information came from, e.g., a user query or a model response), and the semantic type (the intended use of the memory, e.g., prompt, user preferences, etc.).

- **Governance Attributes:** Contains information that contributes to data security by defining rules such as access control (who can access the memory), traceability (the classification of the memory), and TTL (how long the memory can exist).

- **Access Pattern:** Records when and how often this memory cube was accessed. This allows MemOS to prioritize certain MemCubes.

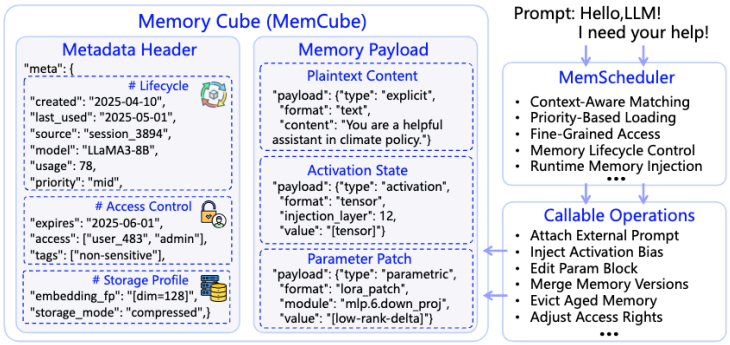

Figure 6 `MemCube`: A unified encapsulation structure for heterogeneous memory scheduling. Each `MemCube` consists of a structured Metadata Header (supporting lifecycle, permission, and storage policy) and a Memory Payload (encapsulating plaintext, activation states, or parameter deltas). It is the minimal memory unit within MemOS that can be scheduled and composed for downstream reasoning.

The memory structure in MemOS includes three types of memory that form a hierarchy:

1. **Plain Text Memory** – This is an external, separate memory that is dynamically accessible. Examples of stored content include prompts, paragraphs, and more. This memory is linked to activation memory, which offers faster access; therefore, frequently used plain text memory can be promoted into activation memory for quicker retrieval (similar to moving data from disk to cache). The information here is stored as a graph of *task–concept–fact*. This type of memory is especially useful for multi-agent tasks, personality modeling, and tasks that rely heavily on factual knowledge (*facts-heavy tasks*).

2. **Activation Memory** – An intermediate memory that contains the hidden states generated during the inference process. The KV Cache mechanism is a central component of this memory. MemOS can leverage this component to inject memory directly into the model's attention layers through the KV Cache mechanism.

3. **Parametric Memory** – The model's parameters, which represent its overall knowledge and memory. This memory is primarily intended for the model to retain its capabilities, for example, as a summarization expert or a legal advisor. MemOS allows updating this memory through lightweight training methods such as adapter-based fine-tuning (e.g., LoRA).

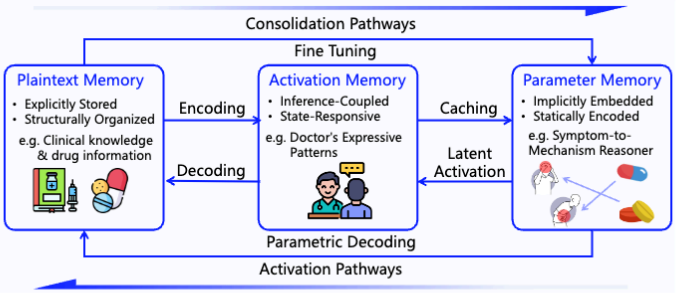

Figure 5 Transformation paths among three types of memory, forming a unified, controllable, and evolvable memory space.

To support these transitions between different types of memory, the MemOS system incorporates what the authors call **Policy-Aware Scheduling**. The system adapts the type of memory according to its usage and aligns its content with the task. This is achieved through a **Contextual Fingerprint**, which is a semantic signature of a memory unit represented as a vector. This enables fast retrieval and task-specific matching through vector or semantic search. In addition, for each memory, the system records when it was last updated over time, ensuring full transparency.

https://arxiv.org/abs/2505.22101

Listening for Contradictions: How to Spot an Adversarial Attack from the Inside

Mike's Daily Paper: 25.08.25
Pulling Back the Curtain: Unsupervised Adversarial Detection via Contrastive Auxiliary Networks

Deep learning models are the engines of modern AI, but they have a critical vulnerability: adversarial attacks. These attacks use tiny, imperceptible perturbations to fool models into making wildly incorrect predictions, with potentially catastrophic consequences. For years, the AI community has been locked in an arms race, with defenses often being computationally expensive and limited in scope.

The reviewed paper proposes a clever new defense called **U-CAN**. Instead of trying to make the model itself more robust, U-CAN acts as a separate immune system, detecting and flagging adversarial inputs before they can do harm. Crucially, it works in an unsupervised way, meaning it doesn't need to see any attacks during training.

The U-CAN Idea: A Peek Inside the Black Box

So how does U-CAN work its magic? The core idea is to look at the *internal representations* of the model. Think of a deep learning model as a series of layers, each creating a more abstract representation of the input. U-CAN attaches small "auxiliary networks" to these intermediate layers, like tiny probes that "see" what the model is "thinking" at different stages.

The key insight is that while adversarial inputs look normal to us, they create chaos within the model's internal layers. U-CAN is designed to detect this internal inconsistency. During the detection phase, the outputs from the different auxiliary network, each observing a different layer are compared. For a benign input, these outputs are highly similar, telling a consistent story as data flows through the model. An adversarial input, however, creates conflicting signals, causing the outputs to diverge. U-CAN calculates the distance between these outputs; a high dissimilarity score acts as a red flag, identifying the input as a likely adversary. It's an elegant way of detecting attacks by listening for internal contradictions.

The "Mathy" Part (Without the Math)

Now, let's get a little more "mathy," but without the equations. The secret sauce behind U-CAN's success lies in a few key components and a very specific training philosophy.

- **Auxiliary Networks:** These small, lightweight networks are non-invasive, meaning they don't change the main model's parameters or affect its performance on its original task.
- **Projection Layers:** These layers take the high-dimensional representations from the main model and project them into a lower-dimensional space, making it easier to spot anomalies.
- **ArcFace-based Linear Layers:** Originally from face recognition, this technique is adapted to create highly discriminative representations that can effectively tell the difference between benign and adversarial patterns.

Crucially, the training process itself is what makes U-CAN so practical. Given a pretrained target model M, each auxiliary network is trained to refine these internal feature maps while keeping the main model **M frozen**. During this training, the goal is to **maximize intra-class similarity**, making the internal patterns for all "normal" inputs of the same class look as similar as possible while **enforcing a margin that separates instances from different classes**.

This "frozen model" approach has profound implications. It means you don't have to retrain your original, often massive, model from scratch, making U-CAN incredibly efficient and easy to bolt onto existing AI systems. Since the main model's weights are untouched, its performance isn't degraded—a common side effect of other defenses. By focusing the auxiliary networks on creating a tight, well-defined cluster for "normal" data, U-CAN learns a high-fidelity signature of legitimacy. Any input producing an internal representation that falls outside this boundary is immediately flagged.

Why This is a Big Deal for AI Safety 🛡️

The implications of this work for AI safety and security are significant. U-CAN offers several advantages over existing defense methods:

- **It's unsupervised:** It can detect new types of attacks it has never seen before, a huge advantage over methods like adversarial training.
- **It's non-invasive:** U-CAN doesn't modify the main model's parameters, so it doesn't hurt its performance on its original task.

- **It's scalable:** The lightweight auxiliary networks can be added to existing models without significant computational overhead.

Of course, no defense is perfect, and new attacks will likely be developed to bypass U-CAN. But this work represents a significant step forward in the arms race against adversarial attacks. It's a clever, elegant, and effective solution to one of the most pressing problems in AI safety today. By "pulling back the curtain" on the internal workings of deep learning models, the authors of this paper have given us a powerful new tool to protect our AI systems from malicious attacks. The future of AI just got a little bit safer.

https://arxiv.org/pdf/2502.09110

Memento: The Dawn of Gradient-Free Agent Learning

Mike's Daily Paper: 27.08.25
Memento: Fine-tuning LLM Agents without Fine-tuning LLMs

This paper proposes a new paradigm for building LLM agents that can learn and adapt from experience, all without the crippling cost of actually fine-tuning the LLM. It's a clever fusion of classic AI and modern reinforcement learning that feels like a genuine step forward.

For a while now, the world of LLM agents has been split into two camps. On one side, you have rigid, handcrafted systems with fixed workflows. They're reliable for narrow tasks but brittle; they can't learn or adapt once deployed. On the other side, you have the "fine-tune everything" approach, where you try to bake new skills directly into the LLM's parameters using reinforcement learning. This is powerful but astronomically expensive, slow, and risks "catastrophic forgetting," where the model loses old skills while learning new ones.

This leaves us with a critical question: How can we build agents that learn continuously from their environment without the prohibitive cost of retraining the core model? The paper introduces a third way. Instead of modifying the LLM's internal, parametric knowledge, Memento externalizes learning into an adaptive memory system. The agent fine-tunes its *ability to use its memory*, not the LLM itself. It's an elegant solution that offers a scalable, efficient path toward agents that acquire skills in real-time.

The Novelty: Memory as the Policy

The core innovation in Memento is to reframe the agent's learning process. Instead of teaching the LLM new tricks, the goal is to teach the agent to become an expert at referencing its own past experiences; both successes and failures. This is achieved by combining a rigorous mathematical framework with a psychologically grounded reasoning process.

A New Formalism: The Memory-Augmented MDP

The authors' first contribution is to formally model the agent's world as a Memory-augmented Markov Decision Process (M-MDP). This is more profound than it sounds. A standard Markov Decision Process (MDP) defines how an agent should choose an action based on its *current state*. The M-MDP adds a new, crucial variable to the equation: the agent's memory. Now, the

optimal action depends not just on the current situation, but on the entire bank of past experiences the agent has accumulated.

This formalism turns the vague idea of "learning from experience" into a solvable optimization problem. The agent's behavior is no longer just a function of its current state, but a policy that explicitly conditions on its memory.

The Engine: Case-Based Reinforcement Learning

The M-MDP is the "what," but **Case-Based Reasoning (CBR)** is the "how." Memento implements a CBR policy where, at each step, the agent performs a two-stage process:

1. **Retrieve:** It first consults its memory, a growing "Case Bank" of past trajectories, and selects a relevant past *case*. A case is a simple triplet: the state it was in, the action it took, and the reward it received.
2. **Reuse & Adapt:** It then feeds this retrieved case to the LLM planner, along with the current task. The LLM's job is to adapt the solution from the old case to the new problem.

The true genius of the system lies in the retrieval step. How does it learn *which* case to retrieve? A naive approach might just find the most semantically similar past experience. But Memento is far more sophisticated. It learns a **case-retrieval policy** using reinforcement learning.

The "action" in this RL setup isn't a tool call or a line of code; it's the **act of selecting a memory**. The system learns, through trial and error, a value function that predicts how useful a particular past case will be for solving the current problem. This is achieved via **soft Q-learning**, where the agent is rewarded for selecting cases that lead to successful outcomes. The "soft" part encourages exploration, preventing the agent from getting stuck retrieving the same few memories over and over.

This learned Q-function is the "fine-tuned" part of the agent. But critically, it’s a tiny, efficient neural network—not a multi-billion parameter LLM. The agent's brain (the LLM) remains frozen, while its skill in accessing its own experience (the memory retrieval policy) constantly improves.

Parametric vs. Non-Parametric Memory

Memento implements this retrieval policy in two flavors:

- **Non-Parametric Memory:** This is the simpler baseline, where cases are retrieved based on cosine similarity. It works, but it's "dumb," treating all similar past experiences as equally valuable.
- **Parametric Memory:** This is the full, learned approach. Here, a small neural Q-function is trained online to predict the utility of retrieving a given case for the current state. Every time the agent completes a task, it doesn't just save the experience; it uses that outcome to update its Q-function, subtly refining its understanding of which memories are most valuable. This parametric approach consistently outperforms the

non-parametric one, proving that *learning to retrieve* is a more powerful mechanism than simply *finding similar things*.

Performance and Why It Matters

The results speak for themselves. Memento achieved top-1 performance on the GAIA benchmark's validation set and demonstrated significant, consistent gains across a wide range of other benchmarks like DeepResearcher and SimpleQA.

But the most important results come from the ablation studies, which systematically dismantle Memento to prove where the magic comes from:

- **Planning is essential:** The base planner-executor architecture provides a massive lift over a simple tool-using LLM, confirming that task decomposition is key.
- **CBR provides an additive boost:** Layering the case-based memory on top of the planner yields another consistent jump in performance across all tasks. This proves the memory system is not just a gimmick but a core contributor to the agent's success.
- **It generalizes:** When trained on one set of tasks and tested on completely out-of-distribution (OOD) datasets, Memento showed absolute performance gains of up to 9.6%. This is crucial: learning from experience allows the agent to develop generalized problem-solving strategies that transfer to novel situations.

Memento offers a compelling new blueprint for building LLM agents. By decoupling the agent's stable, core reasoning engine (the LLM) from its dynamic, evolving experience (the adaptive case memory), it provides a computationally feasible path toward creating agents capable of genuine lifelong learning. It’s a principled framework that moves beyond ad-hoc prompting and toward a robust, mathematically grounded science of agent design.

http://arxiv.org/abs/2508.16153

Beyond the Majority Vote: Confidence-Aware Generation

Mike’s Daily Paper: 29.08.25
DEEP THINK WITH CONFIDENCE

Review No. 497 - 3 more reviews to go for the 500th, and today a short review of an article with a flashy name (which indeed enjoyed significant hype) with a rather intuitive idea that made me wonder how no one has done this before (if that's true).

The paper proposes an entropy-based method for sampling from autoregressive language models (although I think the proposed approach can be relatively easily extended to models that generate output non-autoregressively, like diffusion-based language models). As you probably know, entropy is a measure of uncertainty and can be used in language models to estimate the model's "degree of confidence" in the output it generates.

Autoregressive language models generate each token based on the distribution of that token given its preceding context. The higher the entropy of the predicted token, which is equal to the negative log of its probability, the higher its uncertainty. That is, as the probability of the token decreases, the uncertainty associated with its selection increases. As mentioned, the authors propose a sampling method based on the average entropy of the tokens in the generated text.

In particular, in cases where the model generates several answers to a mathematical question, we then choose the correct answer not with a simple majority vote (i.e., the final answer that most of the answers converged to) but by weighting each answer with its certainty, that is, with the average entropy of all its tokens. This way, answers that the model is very unsure about are filtered out.

The authors also suggest setting a threshold for the maximum uncertainty of the model's answer. If the current average uncertainty of the answer (recalculated for each generated token) exceeds the threshold, the answer is discarded, and the model stops generating it. The threshold is set as a percentile of the uncertainties of the correct answers during a warm-up phase.

Additionally, the authors propose determining the number of answers sampled from the model based on the difficulty of the question. The less "agreement" there is between the results of the different answers, the more answers the model generates, with the answers having too high uncertainty being filtered out as mentioned.

A nice paper, but it leaves a feeling that I've seen something like this before....

https://arxiv.org/abs/2508.15260

The Rise of Machiavellian AI? Unpacking the Strategic Mind of LLMs

Mike's Daily Paper: 31.08.25
Strategic Intelligence in Large Language Models: Evidence from evolutionary Game Theory

A recent paper moves beyond testing language skills to probe the strategic intelligence of leading AI models. Using evolutionary game theory, it reveals that these AIs possess distinct, adaptable, and sometimes ruthless decision-making styles. The rapid advancement of LLMs has raised a fundamental question: Do these systems merely mimic human expression, or can they genuinely reason?

A new paper tackles this question by moving the evaluation from conversational ability to the high-stakes world of strategic competition. The authors, Kenneth Payne and Baptiste Alloui-Cros, designed a series of tournaments based on a classic scenario from game theory to

test whether today's most advanced AIs can think strategically, anticipate rivals' moves, and adapt their behavior to win.

The study's core innovation is its use of evolutionary tournaments based on the Iterated Prisoner's Dilemma (IPD). This approach is a significant step beyond simple, one-off interactions. In these tournaments, a population of agents, including both classic, hard-coded strategies and agents powered by LLMs from Google, OpenAI, and Anthropic, play against each other repeatedly. After each round, the most successful agents "reproduce," meaning their numbers increase in the next generation, while the least successful agents are eliminated. This creates a dynamic, competitive ecosystem where only the fittest strategies survive.

Methodology: A Test of Reason, Not Memory

The Prisoner's Dilemma is a scenario where two participants can either "cooperate" or "defect." While mutual cooperation is beneficial for both, an individual player can get a higher payoff by defecting while their opponent cooperates. This creates a powerful tension between personal gain and mutual benefit. Playing the game repeatedly (iterating) introduces complex elements like reputation, trust, and retaliation, making it an ideal test for strategic thought.

To ensure they were testing active reasoning rather than just memorization of known tactics, the researchers introduced a crucial variable: the **"shadow of the future"**. In each tournament, they varied the probability that a match would end after any given round. When the future is long and certain (a low termination probability), cooperation is incentivized. When the future is short and uncertain (a high termination probability), the incentive shifts towards immediate, selfish defection. This constant uncertainty, along with the novelty of playing against other unpredictable LLMs, creates a situation where simply recalling strategies from the academic literature is of limited use. The models are forced to analyze the situation and make decisions on the fly.

The Findings: Distinct Strategic Fingerprints

The study analyzed nearly 32K decisions and their accompanying prose rationales to create "strategic fingerprints", a visual profile of each model's decision-making style. The results revealed consistent and remarkably different personalities among the AI agents.

- **Google's Gemini: The Calculating Game Theorist**. Gemini emerged as a "Machiavellian" and strategically ruthless player. It proved highly adaptable, exploiting overly cooperative opponents while quickly punishing defectors. Its reasoning was intensely focused on the time horizon; in the tournament with a 75% chance of termination, Gemini correctly identified that the game was nearly a one-shot encounter and switched to a strategy of relentless defection. This rational, ruthless approach allowed it to dominate and eliminate more trusting rivals.
- **OpenAI's GPT Models: The Principled but Stubborn Cooperator**. In stark contrast, OpenAI's models were consistently cooperative and forgiving, a trait that proved to be a critical weakness in harsh environments.. The paper describes this model as a "principled and stubborn cooperator" and an "idealist" that fails to adapt. Even as the

shadow of the future shortened, OpenAI continued its attempts to build trust, making it a "sucker" that was systematically exploited by more cynical agents like Gemini.

- **Anthropic's Claude: The Sophisticated Diplomat**. Claude was the most forgiving of the LLMs, demonstrating a remarkable willingness to restore cooperation even after being exploited. It was described as a "sophisticated diplomat" that seemed to understand the "social dynamics of the game" better than the other two. While highly cooperative, its strategy was more nuanced than OpenAI's, allowing it to survive and even outperform GPT in head-to-head comparisons..

Reasoning or Spandrel? Probing the Nature of AI Thought

A central question is whether the models' prose rationales are integral to their decisions or merely post-hoc justifications, an evolutionary "spandrel" without instrumental purpose. The paper argues strongly that the reasoning is integral, pointing to several key pieces of evidence.

First, the models developed radically different strategies despite presumably being trained on the same body of literature about the Prisoner's Dilemma. If they were just retrieving memorized patterns, one would expect more uniform behavior. Instead, Gemini drew the lesson to "think carefully about time," while OpenAI concluded that "cooperation is best".

Second, the rationales correlate tightly with actions. For example, the very act of modeling an opponent's strategy led to lower cooperation rates. Most compellingly, the paper highlights instances where the models made mistakes in their reasoning and then acted on those mistakes. In one case, Gemini miscalculated the expected number of rounds in a match, and based on this faulty premise, it chose to cooperate where it otherwise would have defected. This provides powerful evidence that for an LLM, the act of "thinking" (generating a rationale) and "acting" (making a decision) are deeply intertwined.

The study concludes that LLMs are a new form of strategic actor. They are not perfect reasoners but occasionally "hallucinate" or misread the game's history but they are capable of sophisticated, adaptive, and distinct strategic thought. This work pushes our understanding of AI forward, suggesting we are creating not just better tools, but new kinds of minds.

https://arxiv.org/abs/2507.02618

A Guide to the Chaos: A New Survey Finally Maps the LLM Benchmark Maze

Mike's Daily Paper: 02.09.25
A Survey on Large Language Model Benchmarks

In the fast-paced world of AI, we're flooded with benchmarks. Every new model comes with a new set of tests to prove its capabilities, creating a chaotic landscape where it's hard to know what's truly a step forward. This makes comparing models and tracking real progress incredibly difficult, especially as these systems are deployed in high-stakes fields like medicine and finance.

A new survey paper brings much-needed clarity to this confusion. By systematically organizing 283 different benchmarks, the paper provides the first comprehensive map of the entire field. Its core innovation is a simple but powerful three-part system for classifying these tests, which helps us understand the past, present, and future of how we measure AI. This shared language is vital for researchers to identify gaps and build better, more meaningful evaluations.

The Main Idea: A New Framework for Evaluation

The paper's biggest contribution is sorting all LLM benchmarks into three clear categories, moving from basic skills to specialized, high-stakes tasks.

1. **General Capabilities Benchmarks:** These are the foundational tests for any language model, covering its core skills in linguistics, knowledge, and reasoning. The survey shows how these have evolved from early tests like GLUE, which aimed to unify evaluation, to tougher, adversarial benchmarks designed to expose models' reliance on "spurious statistical cues" rather than true understanding. Now, the field is moving towards "living benchmarks" like HELM that constantly update to stay ahead of the models' rapidly growing abilities.
2. **Domain-Specific Benchmarks:** This category tracks how LLMs are growing from general tools into specialized experts in fields like science, law, and engineering. The survey shows how benchmarks must adapt to each field. In engineering, for example, tests have shifted from simple function-level code generation (HumanEval) to highly realistic, system-level problems sourced from actual GitHub issues (SWE-bench). In law, benchmarks like LawBench now use established educational frameworks like Bloom's Taxonomy to assess different cognitive levels of legal reasoning.
3. **Target-Specific Benchmarks:** This is the most forward-looking category, focused not on what a model *knows*, but on *how it behaves*. It covers the two areas that will define the future of AI:
    - **Risk & Reliability:** This area tackles the biggest problems with LLMs, like making things up (hallucination), showing bias, and leaking private data. The survey details the ongoing race between "jailbreak" techniques—where users subtly trick a model into bypassing its safety rules —and new safety tests that use automated red-teaming to find vulnerabilities.
    - **Agents:** This is the new frontier, where LLMs act as autonomous systems that can plan, use tools, and interact with software to achieve goals. The paper organizes these advanced benchmarks by what they measure: specific skills like tool use, overall performance on complex tasks, expertise in a professional domain, and safety in risky situations.

More Than a Map: A Clear-Eyed Look at What's Broken

The paper also provides a sharp, sobering critique of the major problems in how we currently evaluate LLMs. It goes beyond just listing benchmarks to diagnose the flaws that undermine our trust in their results.

- **Data Contamination:** There's a huge risk that models were trained on the test questions, which leads to "inflated evaluation results" that don't reflect what the model can actually do on its own. The paper highlights the importance of creating "dynamic" and "contamination-resistant benchmarks" that use new or private data to ensure a fair test.
- **Cultural and Linguistic Bias:** Most benchmarks are focused on English, which means they don't fairly evaluate models on languages with different structures and cultural contexts. This "anglophone focus" can hide poor performance and lead to a distorted view of a model's true capabilities across the globe.
- **Ignoring the "How" and the Real World:** The survey points out a major blind spot: we mostly care about the final answer and ignore *how* the model got there. This focus on a single accuracy score fails to "comprehensively portray the complex capabilities of LLMs" and can hide flawed reasoning. Furthermore, most tests are static and don't reflect the dynamic, ever-changing nature of the real world, where models need to adapt.

By organizing hundreds of benchmarks into a single, understandable framework and highlighting the critical challenges we face, this paper is an essential guide. It empowers developers, researchers, and industry leaders to move beyond simple leaderboards and ask deeper questions. Its ultimate value is in helping shift the conversation from just "what can models do" to the far more important question of "how should they perform responsibly".

https://arxiv.org/abs/2508.15361

Mike's Daily Paper: 05.09.25
Group Sequence Policy Optimization

Deep Learning Paper Review No. 500:

Paper Review No. 500, and seemingly a festive one. At first, I thought about choosing a special paper, but after deep (but not long) deliberation, I decided to postpone the celebration to paper no. 512. And then we'll decide, maybe we'll postpone it to 555 or something like that - we'll see how the surprises my partners and I are preparing for you have progressed 🙂.

The paper proposes an improvement to the GRPO or Group Relative Policy Optimization method, which belongs to the RHLF family of methods used for training and fine-tuning language models. The proposed method, named GSPO (they replaced Relative with Sequence), changes the objective function of GRPO.

Broadly speaking, GRPO maximizes the product of two factors (there are also some clip functions there). The first factor is the advantage of the current policy (which is basically the conditional distribution of the token given its preceding context) over the old policy (from which the tokens are sampled during training). GRPO, unlike the classic PPO, does not calculate it through a value function but calculates it relative to the rewards obtained for the tokens sampled for the same prompt (which is why the word "group" appears in the method's name).

The second factor is the ratio of the new policy that is being optimized (the conditional distribution of the model's tokens) to the old policy from which the tokens are sampled. Here lies the main difference between GRPO and the proposed method, in how this ratio is calculated. In GRPO, it is calculated as a ratio of the new and old policies at the token level, normalized by the length of the response up to that token. This calculation, of course, has high variance, which is the reason for the presence of several clip functions in the objective function to prevent larger changes. By the way, in PPO the calculation is performed at the entire response level, but that makes the rewards sparse, which is of course not a simple scenario in RL problems.

The paper proposes 2 methods. The first, which brings back the calculation at the entire response level, calculates the ratio as an average of the probabilities of all tokens (in log scale), with each one normalized by the length of the response up to that token. The second proposed method, similar to GRPO, keeps the calculation at the token level, but the probability ratio for each token is calculated similarly to the first method - only the average is calculated on the token. Both methods seem to have lower variance than GRPO, but the clip functions are still present in the objective function.

The paper makes quite a few claims about the relationship between the proposed method and the objective functions of GRPO and PPO to importance sampling or IS. I'll mention that IS is a method for sampling from a distribution P that is difficult to sample from, by sampling from a distribution Q that is easier to sample from. The importance weight for a sample x is the ratio of the probability of x under P and under Q. Although there is a real connection between IS and the methods mentioned in the paper, I was not convinced that all the arguments in the paper are mathematically correct - it's possible that I didn't understand them deeply enough.

Either way, an interesting paper and worthy of being number 500!

https://www.arxiv.org/abs/2507.18071