Why & How Does Few-Shot Learning Work?

Transformers as Learning Algorithms

Jörn Stöhler (MSc Student) Claude (Research Assistant)

University of Augsburg

What is Few-Shot Learning?

Live Demonstration

Demo in ChatGPT.com

1. Pattern completion: `"The cat sat on the mat.  The dog sat on the..."`
2. Zero-shot fails, few-shot succeeds
3. Learning notation from examples

Key Question: Examples transform behavior – but how?

Quantifying the Effect

Benchmarks from GPT-3 Paper

Open: GPT-3 Paper (Brown et al. 2020)

`papers/2005.14165_gpt3_language_models_few_shot.pdf`

Show:

- Figure 1.2: Performance vs parameters
- Figure 3.1: Zero/one/few-shot visual
- Figure 3.8: LAMBADA (76.2% $\rightarrow$ 86.4%)

Model	Year	Zero-Shot	Few-Shot Gain
GPT-3	2020	~50%	+10-20pp
GPT-4	2023	~80%	+2-8pp
Current	2024	~85%	+1-5pp

SuperGLUE Results

Few-Shot vs Fine-Tuning

Open: GPT-3 Paper - SuperGLUE

`papers/2005.14165_gpt3_language_models_few_shot.pdf`

Key Result:

- GPT-3 (32-shot): 71.8% (average)
- Fine-tuned BERT++: 69.8% (average)
- **No gradient updates needed!**

Transition: The effect is real. Now let's understand the mechanism.

Transformer Architecture

Information Flow

$$\text{Tokens} \longrightarrow \underset{a_0 \in \mathbb{R}^d}{\text{Embeddings}} \longrightarrow \underset{a_1 = f_1(a_0)}{\boxed{\text{Layer 1}}} \longrightarrow \cdots \longrightarrow \underset{a_L}{\boxed{\text{Layer L}}} \longrightarrow \text{Output}$$

$$\text{Attention}(a) = \sum_t \text{softmax}(Q_t K_t^T) \cdot V_t$$

Key Points:

- Residual stream: Information highway
- Each layer reads ALL previous tokens
- Autoregressive: One token at a time

Open: von Oswald et al. 2022
papers/2212.07677_transformers_gradient_descent.pdf

von Oswald et al. 2022 - Key Result

For linear self-attention on regression:

$$\text{Output} = \text{Input} + \eta \cdot \nabla_w \mathcal{L}$$

Linear attention = exact gradient descent step

- Single layer = one gradient step (exact!)
- Multi-layer = preconditioned gradient descent
- Not approximation – mathematically exact

What We've Found Inside

Mechanistic Interpretability

1. **Pattern Completion**
   - Attention patterns copy previous tokens
   - Demonstrated in GPT models
   - *Note: Anthropic details online only*
2. **Function Vectors**
   - Tasks encoded as activation directions
   - Todd et al. 2024
   - Can extract and compose task vectors
3. **Multiple Capabilities**
   - Models encode many tasks simultaneously
   - Context selects relevant circuits
   - Examples activate specific pathways

Layer-Wise Processing

Information Flow Through Depth

Papers: Akyürek 2022 + Garg 2022
2211.15661 + 2208.01066

Akyürek et al. findings:

- 1 layer: Single gradient descent step
- 2-3 layers: Multiple GD steps
- 4+ layers: Ridge regression emerges
- Deep models: Approach closed-form solutions

Exact depth depends on task complexity

What Can Transformers Provably Do?

Mathematical Limits

What Papers Actually Prove:

1. **Linear attention = GD** – von Oswald 2022
2. **Can implement ridge** – Akyürek 2022
3. **Learn linear functions** – Garg 2022
4. **Hard attention is Turing complete** – Pérez 2019

Recent Result (2024):

- Prompting itself is Turing-complete
- Paper: Prompting is Turing-complete (2024)

Memory Bounds:

- $\Theta(n)$ capacity for n examples
- Tian et al. 2024

Mesa-Optimization

Two Optimization Loops

Papers (both needed):

[1] Hubinger 2019 - Terminology

[2] von Oswald 2023 - Evidence

[3] Zheng 2024 - Emergence

	OUTER (Training)	**INNER (Inference)**
Optimizer	SGD on parameters θ	Attention implements GD
Objective	Training loss	In-context loss
Updates	Weights	Activations
Time	Months	Single forward pass

Critical insight: Model learns HOW to learn, not just WHAT to predict

Emerges without design! Never explicitly trained for optimization

Mesa-Optimization Evidence

Internal Optimizer Discovery

Open: Uncovering Mesa-Optimization

`papers/2309.05858_mesa_optimization.pdf`

Two-stage process discovered:

1. **Early layers:** Preconditioning
2. **Later layers:** Optimization on preconditioned problem

Key findings:

- Autoregressive training $\rightarrow$ internal optimizers
- Generalizes to unseen tasks
- Can extract the learned algorithm

Why Few-Shot Works

The Grad Student Analogy

Few-shot learning $\approx$ Supervising grad students

1. **Examples provide new information**
   - Your specific notation
   - Not in training data
2. **Computable format**
   - Examples $>$ descriptions
   - Model runs gradient descent on them
3. **Disambiguate task**
   - "Prove like Bourbaki, not Arnold"
4. **Knowledge loading (push system)**
   - Examples activate circuits
   - Weights $\rightarrow$ activations

Examples are training data for the internal optimizer!

Conclusion

The Key Insight

Few-shot learning works because transformers are computers that run learning algorithms

- Your examples are the **program**
- The forward pass is the **execution**
- The output is the result of **internal optimization**

This isn't metaphorical – it's mathematically proven

Questions?

15-minute Q&A – Note: Some claims span multiple papers

Appendix Topics Available:

- A1: Detailed mechanistic interpretability
- A2: Statistical learning theory connection
- A3: Test-time training advances
- A4: Prompt engineering theory
- A5: Failure modes and limitations

Key Papers (all in papers/ folder):

- von Oswald 2022 – Gradient descent proof
- von Oswald 2023 – Mesa-optimization
- Akyürek 2022 – Algorithm identification
- Garg 2022 – What transformers learn
- Hubinger 2019 – Mesa terminology

A1: Mechanistic Interpretability Details

Circuit Discovery

Induction Heads:

- Previous token head + induction head
- Implements $[A][B] \ldots [A] \rightarrow [B]$
- Anthropic (2023) - web article

Sparse Autoencoders:

- Extract monosemantic features
- Reveals superposition
- Anthropic (2024) - scaling study

Knowledge Circuits:

- Early: Query formation
- Middle: Knowledge retrieval
- Late: Answer formatting

A2: Statistical Learning Theory

Generalization Guarantees

Generalization Theory:

- Transformers satisfy PAC-learning bounds
- Algorithmic stability provides guarantees
- *Li et al. 2023 (ICML proceedings)*

Rademacher Complexity:

- Sequence-length independent bounds
- Explains why models don't overfit to examples
- Classical theory (VC dimension, Rademacher)

Statistical Optimality:

- Achieves minimax rates for regression
- 2024 paper

A3: Test-Time Training

Explicit Optimization at Inference

Paper: Test-Time Training (2025)

Idea: Gradient updates on context examples during inference

Benefits:

- Combines parametric + non-parametric learning
- Better sample complexity
- Provable improvements

Connection: Makes mesa-optimization explicit!

A4: Prompt Engineering Theory

Optimal Example Selection

Example Selection Theory:

- Coverage vs similarity tradeoff
- Order matters for performance
- *Multiple papers on selection strategies*

Order Effects:

- Entropy ordering works best
- 17-point improvement on compositional tasks
- Survey paper

OPRO (LLMs as Optimizers):

- Natural language optimization
- 50% improvement on reasoning
- Yang et al. 2023

A5: Failure Modes

When Few-Shot Doesn't Help

Limitations:

1. **Context window constraints**
   - Memory: $\Theta(n)$ for n examples
2. **Task misalignment**
   - Mesa-objective $\neq$ your objective
3. **Distribution shift**
   - Examples not representative
4. **Adversarial examples**
   - Can hijack internal optimizer

When it fails:

- Novel capabilities not in training
- Contradictory examples
- Tasks requiring true reasoning (not pattern matching)

Additional Resources

For Further Reading

Core Papers:

- GPT-3 – Original few-shot benchmarks
- GPT-4 – Modern performance
- GPT-Fathom – Model comparisons

Theory Papers:

- Universal approximation
- Turing completeness
- Transformers as statisticians

Practical Guides (web):

- PromptHub few-shot guide (see papers/*.md)
- Analytics Vidhya comparison (see papers/*.md)