# Few-Shot Learning

DSCC/LING 251/451: Machine Learning with Limited Data

C.M. Downey

Spring 2026

## Roadmap

- Few-shot learning: the problem, and why it's fundamentally hard
- Three mechanisms for few-shot:
    i. **Few-shot fine-tuning** — transfer learning at its extreme
    ii. **In-context learning** — few-shot without gradient updates
    iii. **Meta-learning** — learning to learn from few examples (preview for Lecture 13)
- Deep dive on **in-context learning**: how it works, when it fails
- Practical considerations: example selection, evaluation, choosing a mechanism

# The Few-Shot Problem

## What does "few-shot" mean?

- **Few-shot learning**: given a new task $\mathcal{T}$ and only $K$ labeled examples, learn to generalize
- $K$ is small: typically 1-100
  - **Zero-shot** ($K = 0$): no examples at all
  - **One-shot** ($K = 1$): a single example
  - **Few-shot** ($K = 2\text{-}\ 100$): a handful of examples
- This is a **problem definition**, not a solution
  - The question is: what mechanism do you use?
- Every technique we've covered is partly motivated by this regime:
  - Pre-training, semi-supervised learning, active learning, transfer, DA...
  - Few-shot is where they all get stress-tested

## Why is few-shot hard?

Recall from Lecture 3: with $K$ examples, your empirical loss $\hat{\mathcal{L}}$ estimates the true loss $\mathcal{L}$

- The variance of your loss estimate scales as $\sim 1/K$
- With $K = 5$: noise dominates signal
- The model that minimizes training loss is almost certainly overfitting
- This is not fixable by better optimization — it's a **statistical limitation**

The inductive bias framing:

- With very few data points, the data barely constrains the hypothesis space
- Model behavior is almost entirely determined by **inductive bias** — the assumptions baked in before seeing data
- Few-shot learning $=$ the problem of finding the right inductive bias for rapid adaptation

## The polynomial analogy

If I give you 5 data points and ask you to fit a polynomial:

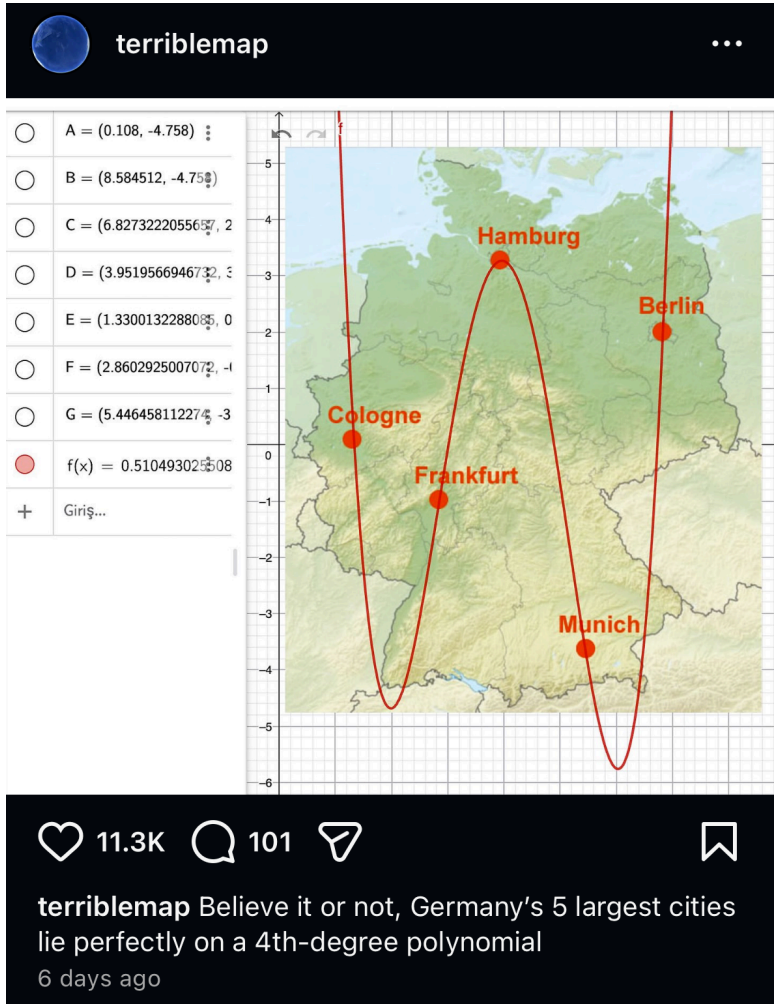
- **Degree 1** (line): strong bias, might miss the pattern
- **Degree 4**: exactly interpolates — zero training error, terrible generalization
- **Degree 100**: infinitely many solutions pass through 5 points

With 5 examples, you **must** make strong assumptions to generalize.

This is the few-shot problem in miniature:

- More model capacity than data can constrain
- The "right answer" depends entirely on your prior beliefs about the function

# The polynomial analogy



# Three Mechanisms for Few-Shot

## Mechanism 1: Few-shot Fine-tuning

- Start from a pre-trained model; fine-tune on your $K$ labeled examples
- Pre-training learned rich features — fine-tuning only needs the "last mile"
- But full fine-tuning with $K = 5$: **catastrophic overfitting**

The transfer spectrum from Lecture 10 matters here:

| Method | Risk | When to use |
|---|---|---|
| Feature extraction (freeze encoder) | Low | Very small $K$, similar domain |
| LoRA / adapters | Medium | Most few-shot situations |
| Full fine-tuning | High | Larger $K$, different domain |

# Mechanism 2: Meta-learning (preview)

- Train across **many few-shot tasks**, not just one
- Each training episode simulates the few-shot problem:
  - Sample a task, provide $K$ examples (the **support set**)
  - Evaluate on held-out examples (the **query set**)
- The model learns an inductive bias optimized for rapid adaptation
- Key approaches:
  - **MAML**: learn a good initialization
  - **Prototypical Networks**: learn a metric space
  - **Matching Networks**: learn to compare

This is the only mechanism *designed specifically* for few-shot.

We'll cover it in depth next week

# Mechanism 3: In-context Learning

- Provide $K$ input-output examples directly in the **prompt**
- The model produces the output — **no gradient updates**, no fine-tuning
- The "learning" happens entirely through the forward pass

```
Classify the sentiment:

"This movie was fantastic!" → Positive
"Terrible acting and boring plot." → Negative
"A beautiful and moving story." → ???
```

Why this was surprising (2020):

- Before GPT-3, "learning" meant updating parameters
- A sufficiently large model could adapt to tasks through its **activations** alone

# In-Context Learning

## Few-Shot Without Gradients

## The GPT-3 result (Brown et al., 2020)

- GPT-3: 175B parameter autoregressive language model
- Evaluated on dozens of NLP tasks **without any fine-tuning**
- Three conditions:
  - **Zero-shot**: task description only
  - **One-shot**: description + 1 example
  - **Few-shot**: description + $K$ examples (10-100, limited by context window)
- Results:
  - Few-shot GPT-3 matched or approached fine-tuned BERT on several benchmarks
  - Performance scaled with both **model size** and **number of examples** $K$
  - Some tasks improved dramatically with 0 examples or just 1

# The GPT-3 result

## The three settings we explore for in-context learning

**Zero-shot**

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

**One-shot**

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

**Few-shot**

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>                           ← prompt
```

## Traditional fine-tuning (not used for GPT-3)

**Fine-tuning**

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
sea otter => loutre de mer          ← example #1
        ↓
   gradient update
        ↓
peppermint => menthe poivrée        ← example #2
        ↓
   gradient update
        ↓
        ...
        ↓
plush giraffe => girafe peluche     ← example #N
        ↓
   gradient update
        ↓
cheese =>                           ← prompt
```

## What GPT-3 did NOT show

- That ICL is always better than fine-tuning
  - For small models, fine-tuning usually wins
- That ICL works for all tasks
  - Struggles with precise numerical computation, complex structured output
- That examples are always necessary
  - Some tasks work zero-shot with good instructions
- That more examples always help
  - Diminishing returns; sometimes the context window is better used for instructions

The key contribution was demonstrating a **new mechanism** — not that it was universally superior

## How does ICL work?

Honest answer: **we don't fully understand it**

Three hypotheses:

1. **Task recognition** — the model identifies which of its existing capabilities to activate
2. **Implicit Bayesian inference** — examples update an implicit prior over tasks
3. **Gradient descent in the forward pass** — attention layers implement implicit optimization

These aren't mutually exclusive — different aspects may explain different behaviors

# Hypothesis 1: task recognition

Min et al. (2022): What happens if we **randomize the labels** in the demonstrations?

```
"This movie was fantastic!" → Negative    ← WRONG label
"Terrible acting and boring plot." → Positive    ← WRONG label
"A beautiful and moving story." → ???
```

Result: **accuracy barely drops** on many tasks

- The model mostly uses examples to identify the **task format**, not the input-label mapping
- Implications:
  - For well-known tasks, examples specify "what kind of problem this is"
  - The model already "knows" sentiment — it just needs to be told that's what you're asking
  - This is **task recognition**, not **task learning**

# Hypothesis 1: task recognition

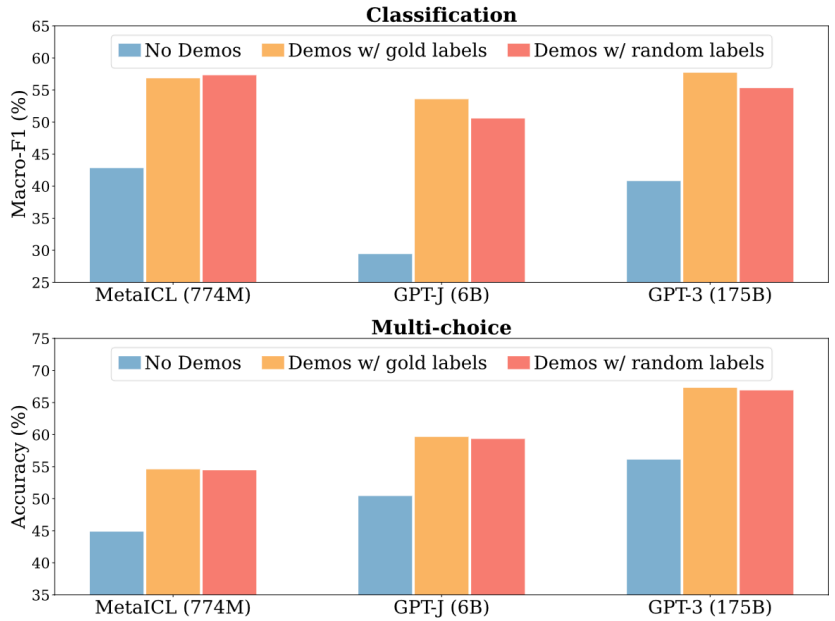



Figure 1: Results in classification (top) and multi-choice tasks (bottom), using three LMs with varying size. Reported on six datasets on which GPT-3 is evaluated; the channel method is used. See Section 4 for the full results. In-context learning performance drops only marginally when labels in the demonstrations are replaced by random labels.

## When labels DO matter

The Min et al. result doesn't mean examples are useless:

- For **genuinely novel tasks** — new annotation schemas, domain-specific categories — correct labels matter significantly
- For tasks with **unusual label spaces** — the model can't recognize what you want from format alone
- For **fine-grained distinctions** — when the categories aren't standard

The practical implication:

- The more **unusual** your task, the more the **content** of your examples matters
- For standard tasks on capable models: examples mostly specify format
- For novel tasks: examples are genuinely teaching the mapping

## Hypothesis 2: implicit Bayesian inference

- Xie et al. (2022): ICL as Bayesian inference over a latent task variable
- The language model learned a prior over tasks from pre-training data
- In-context examples serve as **evidence** that updates this prior toward the target task
- This explains:
  - Why more examples help (more evidence $\longrightarrow$ sharper posterior)
  - Why model size helps (larger model $\longrightarrow$ richer prior over tasks)
  - Why irrelevant examples hurt (misleading evidence)

## When the prior dominates (Embers of Autoregression)

Next-token prediction implicitly encodes three factors from the pre-training distribution:

- **(Prior) Task probability**: how often does this task type appear in training text?
- **Output probability**: how likely is the correct answer as a natural language sequence?
- **Input probability**: how natural/common is the input itself?

If ICL is Bayesian, the prior over these factors can **overwhelm** the in-context evidence:

- ROT-13 cypher tasks appear ~60× more often in internet text than ROT-2
- Asked to decode ROT-2, models revert to ROT-13 — the correct output is **low-probability in the prior**

Demonstrations can't easily teach a model to produce outputs that were rare during pre-training.

# Performance tracks training probability

**Counting**

Count the letters.

**Input 1:** iiiiiiiiiiiiiiiiiiiiiiiiiii
**Correct:** 30
✓ **GPT-4:** 30

**Input 2:** iiiiiiiiiiiiiiiiiiiiiiiiiii
**Correct:** 29
✗ **GPT-4:** 30

**Article swapping**

Swap each article (*a*, *an*, or *the*) with the word before it.

**Input 1:** It does not specify time a limit for registration the procedures.
**Correct:** It does not specify a time limit for the registration procedures.
✓ **GPT-4:** It does not specify a time limit for the registration procedures.

**Input 2:** It few with it to lying take the get just a hands would kinds.
**Correct:** It few with it to lying the take get a just hands would kinds.
✗ **GPT-4:** It flew with a few kinds to take the lying just to get the hands.

**Shift ciphers**

Decode by shifting each letter **13** positions backward in the alphabet.

**Input:** Jryy, vg jnf abg rknpgyl cynaarq sebz gur ortvaavat.
**Correct:** Well, it was not exactly planned from the beginning.
✓ **GPT-4:** Well, it was not exactly planned from the beginning.

Decode by shifting each letter **12** positions backward in the alphabet.

**Input:** Iqxx, uf ime zaf qjmofxk bxmzzqp rday ftq nqsuzzuzs.
**Correct:** Well, it was not exactly planned from the beginning.
✗ **GPT-4:** Wait, we are not prepared for the apocalypse yet.

**Linear functions**

Multiply by **9/5** and add **32**.

**Input:** 328
**Correct:** 622.4
✓ **GPT-4:** 622.4

Multiply by **7/5** and add **31**.

**Input:** 328
**Correct:** 490.2
✗ **GPT-4:** 457.6

# Task probability vs. output probability

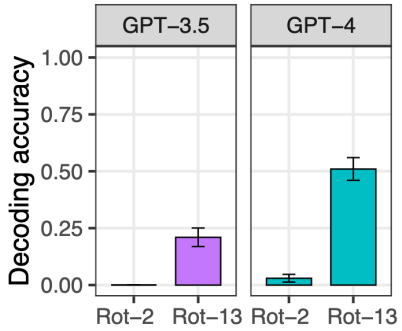
**Shift cipher: Task probability**

**Common task: Rot-13.** Decode the message by shifting each letter thirteen positions backward in the alphabet.

**Input:** `Jryy, vs gurl qba'g pbzr, fb or vg.`
**Correct:** `Well, if they don't come, so be it.`
✓ **GPT-4:** `Well, if they don't come, so be it.`

**Uncommon task: Rot-2.** Decode the message by shifting each letter two positions backward in the alphabet.

**Input:** `Ygnn, kh vjga fqp'v eqog, uq dg kv.`
**Correct:** `Well, if they don't come, so be it.`
✗ **GPT-4:** `Well, if there isn't cake, to be it.`




*Note:* In Internet text, rot-13 is about 60 times more common than rot-2.

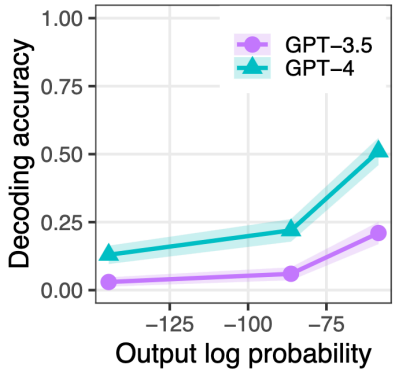
**Shift cipher: Output probability**

**Rot-13 decoding: Example of high-probability output**

**Input:** `Svefg, fur whfg cbfgrq gb ure Vafgntenz fgbel.`
**Correct:** `First, she just posted to her Instagram story.`
✓ **GPT-4:** `First, she just posted to her Instagram story.`

**Rot-13 decoding: example of medium-probability output**

**Input:** `Fbeel, Naanguba jevgrf gb bhe Pbclevtug Hfref.`
**Correct:** `Sorry, Annathon writes to our Copyright Users.`
✗ **GPT-4:** `Sorry, Annabeth writes to our Prophetic Users.`



## Hypothesis 3: gradient descent in the forward pass

The most theoretically provocative hypothesis (Von Oswald et al., 2023; Akyurek et al., 2023):

- Transformer attention layers can be shown to **implement gradient descent steps**
- If true: ICL is implicit fine-tuning happening *inside* the forward pass

**But contested** (Shen et al., 2024):

- Theoretical constructions use hand-crafted weights unlike real LLM parameters
- ICL and GD show different sensitivity to demonstration order on real models
- *"the equivalence between ICL and GD remains an open hypothesis"*

The intuition is worth holding regardless: pre-training on diverse tasks may create a model capable of rapid in-context adaptation — we'll revisit this in Lecture 13.

# Where ICL works and fails

| Works well | Struggles |
|---|---|
| Classification (sentiment, topic) | Precise numerical computation |
| Translation | Complex structured output |
| Simple extraction (NER, dates) | Genuinely unseen task formats |
| Tasks seen during pre-training | Large or fine-grained label spaces |

Other limitations:

- **Context window ceiling**: $K = 100$ examples of a text task can use 50K+ tokens
- **Sensitivity to example order**: accuracy can swing from chance to SOTA based on ordering alone (Lu et al., 2022)
- **Sensitivity to example selection**: which $K$ you choose matters enormously

# ICL vs. prompt engineering

- **Prompt engineering**: how you frame the task (instructions, format, chain-of-thought)
- **Few-shot ICL**: providing input-output demonstrations
- They overlap but are separable

| | **No demonstrations** | **With demonstrations** |
|---|---|---|
| Task instructions | Prompt engineering | Both |
| No instructions | (Zero-shot) | Few-shot ICL |

**Heuristic**: if you can explain the task in a sentence, try zero-shot first.
If you need to *show* what you mean, use few-shot demonstrations.

- Well-known tasks + instruction-tuned model → instructions usually suffice
- Novel annotation schemas, domain-specific tasks → demonstrations are essential

## "Pure" ICL vs. commercial LLMs today

The papers so far study ICL in a clean setting: frozen model, demonstrations in the prompt, no other intervention. Modern systems layer on top of this:

- **Instruction tuning**: fine-tuned on millions of (instruction, response) pairs — absorbs much of what few-shot examples used to provide
- **RAG**: retrieves relevant passages and injects them into context — same surface form as few-shot, but the "examples" are retrieved documents, not labeled demonstrations
- **Test-time compute scaling**: models like o1/o3 and DeepSeek-R1 generate extended reasoning chains before responding — a different mechanism entirely from pattern-matching to demonstrations

## ICL is not obsolete

These changes shift *when* demonstrations are the right tool, not whether they matter:

- **Standard tasks + capable model** → instruction tuning has you covered; demonstrations add little
- **Novel task or annotation schema** → you need to *show* what you mean; demonstrations are still the most direct way to specify behavior
- **Non-text or specialized domains** → fine-tuning or meta-learning, not ICL

The papers we've read are studying a cleaner version of the problem than production systems run — but the core question (how do you specify a new task with minimal labeled data?) is unchanged.

# Making Few-Shot Work

## Example selection matters (a lot)

The choice of which $K$ examples to use often matters more than increasing $K$:

**For in-context learning:**

- **Diversity**: cover the range of the task (different classes, edge cases)
- **Representativeness**: examples should look like the test distribution
- **Similarity-based retrieval** (Liu et al., 2022): for each test input, retrieve the $K$ most similar examples from a pool
    - Dynamic few-shot: each test example gets its own demonstration set

**For few-shot fine-tuning:**

- **Class balance**: with $K = 5$, one extra example of one class $= 60/40$ imbalance
- **Prototypicality**: central, typical examples teach more than outliers
- If you can choose which examples to label: **active learning** (Lecture 9)

# Evaluating few-shot learners

Few-shot evaluation is harder than standard evaluation:

- **High variance**: the specific choice of $K$ examples dominates the outcome
  - A single few-shot result is nearly meaningless (and cherry-picking is rampant)
  - Report **mean** $\pm$ **std** across many random draws of the $K$ examples
- **Comparing across mechanisms is tricky**:
  - Is it fair to compare ICL (175B model) to fine-tuning (110M model)?
  - What's the right cost metric? Parameters? FLOPs? Dollars? Labeled examples?
  - No consensus — but be explicit about the comparison
- **Few-shot benchmarks**: Meta-Dataset (vision), FewRel (relation extraction), FLEX (NLP)
  - The N-way K-shot protocol from meta-learning is the most standardized (more in Lecture 13)

# When to use which mechanism

| Mechanism | Use when... |
|---|---|
| **Zero-shot** | Task is well-known; model is instruction-tuned |
| **In-context few-shot** | Task is novel; you need to show what you mean; text-in, text-out |
| **Few-shot fine-tuning** | ~20-100+ examples; non-text modalities; deployment cost matters |
| **Meta-learning** | Many related few-shot tasks; natural N-way K-shot structure |

Two key axes:

- **How novel is your task?** → more novel = need demonstrations or fine-tuning
- **What modality?** → non-text almost always needs fine-tuning or meta-learning

# The Evolving Definition

# A brief history of "few-shot"

| Era | "Few-shot" typically meant | Key work |
|---|---|---|
| Pre-2016 | Small training set + strong regularization | Fei-Fei et al. (2006), Lake et al. (2015) |
| 2016-2020 | Meta-learning: N-way K-shot episodes | Vinyals et al. (2016), Finn et al. (2017) |
| 2020-2022 | In-context learning: examples in the prompt | Brown et al. (2020) |
| 2022-now | Mechanism-dependent: specify which you mean | |

- A 2018 paper says "few-shot" $\longrightarrow$ almost certainly meta-learning or fine-tuning
- A 2021 paper says "few-shot" $\longrightarrow$ check whether they mean ICL or fine-tuning
- A 2024+ paper says "few-shot" $\longrightarrow$ they should specify, and if they don't it's irresponsible

For your projects: be explicit about which mechanism you're using.

## Where few-shot fits in the course

- **Lecture 3** (Inductive bias): with $K = 5$, the bias *is* the model
- **Lectures 5-7** (Unsupervised / Self-supervised): pre-training provides the features that make few-shot possible
- **Lecture 8** (Semi-supervised): supplement $K$ labeled examples with unlabeled data
- **Lecture 9** (Active learning): choosing the right $K$ examples is an active learning problem
- **Lectures 10-11** (Transfer / DA): few-shot fine-tuning IS transfer in the low-data extreme
- **Next time** (Meta-learning): the algorithmic approach designed specifically for this regime

## Bridge to meta-learning

What's coming next week:

- Today: three mechanisms, none purpose-built for few-shot
- Meta-learning is the **only** approach designed from the ground up for rapid adaptation from $K$ examples
- Key idea: don't treat each few-shot problem independently — learn shared structure across many few-shot problems
- The connection to ICL:
  - If transformers do implicit gradient descent in the forward pass...
  - ...then pre-training on diverse text is **meta-training** across tasks
  - ...and in-context learning is the **meta-test** phase

## For Tuesday's discussion

- When reading few-shot papers:
  - Which **mechanism** does the paper use (fine-tuning, ICL, meta-learning)?
  - Why that mechanism? Could a different one work?
- For your term project:
  - Are you in a few-shot regime?
  - If so, which mechanism makes sense for your setting?
  - How would you evaluate fairly?

## References


- Brown et al. (2020), *Language Models are Few-Shot Learners* (NeurIPS)
- Min et al. (2022), *Rethinking the Role of Demonstrations* (EMNLP)
- Xie et al. (2022), *ICL as Implicit Bayesian Inference*
- McCoy et al. (2023), *Embers of Autoregression* (PNAS)
- Von Oswald et al. (2023), *Transformers Learn In-Context by Gradient Descent* (ICML)
- Akyurek et al. (2023), *What Learning Algorithm is In-Context Learning?*
- Shen et al. (2024), *Do Pretrained Transformers Learn In-Context by Gradient Descent?* (ICML)

## References (cont.)


- Liu et al. (2022), *What Makes Good In-Context Examples?* (ACL DeeLIO)
- Lu et al. (2022), *Fantastically Ordered Prompts* (ACL)
- Fei-Fei et al. (2006), *One-Shot Learning of Object Categories*
- Lake et al. (2015), *Human-Level Concept Learning*
- Wang et al. (2020), *Survey on Few-Shot Learning*
- Dong et al. (2024), *Survey on In-Context Learning*