BEAG LABS

ML Cookbook 2026

52 MODERN TRAINING RECIPES
EVERY AI ENGINEER SHOULD KNOW

A practical collection of state-of-the-art training recipes across 7 domains: language models, vision, 3D generation, speech, robotics, agents, and synthetic data.

LANGUAGE MODELS · VISION · 3D GENERATION · SPEECH · ROBOTICS · AGENTS · SYNTHETIC DATA

B_

Contents

PART 1

Language Models

12 recipes progressing from reward-based RL methods (GRPO, DAPO, RLVR) through alignment techniques (Preference Optimization, Constitutional AI, Process Supervision) to self-improving systems (Recursive Self-Improvement, Synthetic Curriculum, Interaction Models) and alternative architectures (State-Space Models, Linear Attention).

RECIPE 01 | PART LANGUAGE-MODELS

Group Relative Policy Optimization

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 8–64 H100 GPUs for 1–3 days depending on model size and group size

PURPOSE

Improve reasoning capabilities through group-based advantage estimation, eliminating the need for a separate value/critic network.

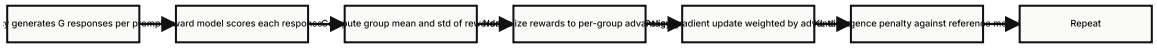
USED BY

DEEPSEEK R1 | DEEPSEEK R1-ZERO | FRONTIER REASONING MODELS

CORE IDEA

Instead of training a separate value network to estimate advantages, GRPO samples a group of responses from the policy, scores each response with a reward model, and computes advantages relative to the group mean. The policy is then updated to increase the probability of responses that scored above average. A KL penalty keeps the policy from drifting too far from the reference model.

TRAINING PIPELINE



ADVANTAGES

- No critic/value network needed — reduces memory and compute
- Lower variance than REINFORCE due to group-based baselining
- Naturally explores diverse responses via group sampling
- Simpler training pipeline than PPO

DISADVANTAGES

- Requires large per-prompt batch sizes (G = 8–64) for stable advantages
- Reward model must be fast enough to score all responses in group
- Group size is a sensitive hyperparameter

BEST FOR

MULTI-STEP REASONING (MATH, CODE, LOGIC)

TASKS WITH VERIFIABLE OUTCOMES

SCENARIOS WHERE YOU HAVE A GOOD REWARD MODEL

KEY PAPERS

[1] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning — DeepSeek AI, 2025

[2] DeepSeekMath: Pushing the Limits of Mathematical Reasoning — DeepSeek AI, 2024

[3] The N+ Implementation Details of RLHF with PPO — Huang et al., 2025

VARIANTS

- DAPO (decoupled clip + dynamic sampling)
- GRPO with length penalty for concise reasoning

OPEN SOURCE

`OpenRLHF (https://github.com/OpenRLHF/OpenRLHF)`

`veRL (https://github.com/volcengine/verl)`

`TRL (https://github.com/huggingface/trl)`

COMMON MISTAKES

- *Setting group size too small (G < 4) leads to high advantage variance*
- *Not normalizing rewards within each group (absolute rewards create bias)*
- *Over-weighting the KL penalty, preventing meaningful policy improvement*

FUTURE DIRECTIONS

Adaptive group sizing, heterogeneous group sampling (different model sizes in same group), and GRPO for multi-modal reasoning.

RECIPE 02

Decoupled Clip and Dynamic Sampling Policy Optimization

COMPLEXITY

★★★★☆

COMPUTE

Medium — 8–64 H100 GPUs for 2–5 days (longer due to sample filtering overhead)

PURPOSE

Address GRPO stability and efficiency issues by decoupling the clipping mechanism and dynamically filtering samples that contribute noise to the policy gradient.

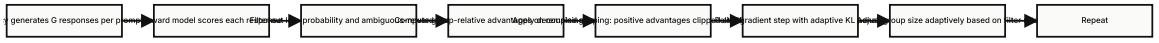
USED BY

BYTEDANCE (SEED1.5-VL) | LARGE-SCALE REASONING SYSTEMS

CORE IDEA

DAPO improves on GRPO with two key modifications. First, it decouples the clipping of positive and negative advantages so that positive updates are not constrained by the same clip threshold as negative ones — allowing the model to reinforce good responses more aggressively. Second, it dynamically filters out samples with very low probability under the current policy (outdated samples) and samples where the reward signal is ambiguous, reducing gradient noise.

TRAINING PIPELINE



ADVANTAGES

- More stable training than GRPO by removing noisy gradient contributions
- Decoupled clipping allows faster learning from positive examples
- Adaptive batch sizing reduces wasted computation on low-quality samples
- Better sample efficiency than vanilla GRPO

DISADVANTAGES

- More hyperparameters than GRPO (two clip epsilons, filter thresholds)
- Filtering can discard hard but informative examples if thresholds are wrong
- Requires careful tuning of the adaptive group size mechanism

BEST FOR

LARGE-SCALE RL TRAINING (1000+ GPU HOURS)

VISION-LANGUAGE REASONING TASKS

SETTINGS WHERE GRPO SHOWS TRAINING INSTABILITY

VARIANTS

- Soft filtering (weighting samples by quality instead of binary discard)
- DAPO with length reward shaping for concise reasoning

KEY PAPERS

[1] DAPO: An Open-Source Framework for Distributed Large-Scale Reinforcement Learning — ByteDance Seed Team, 2025

[2] seed1.5-vl: Pioneering the Path to Multimodal Reasoning — ByteDance Seed Team, 2025

OPEN SOURCE

`veRL (https://github.com/volcengine/verl)`

`OpenRLHF (https://github.com/OpenRLHF/OpenRLHF)`

COMMON MISTAKES

- *Setting the clip threshold for positive advantages too high (reward hacking)*
- *Over-filtering: discarding > 50% of samples reduces effective batch size too much*
- *Not annealing the filter rate as training converges*

FUTURE DIRECTIONS

Learn-to-filter: train a separate gating network to decide which samples to keep, rather than using hand-crafted heuristics.

RECIPE 03

On-Policy Distillation

COMPLEXITY

★★★★☆

COMPUTE

High — requires teacher inference for each student rollout; 16–128 GPUs for 1–2 weeks

PURPOSE

Improve a student model by having it generate responses and then learn from teacher corrections of those trajectories, reducing exposure bias while minimizing teacher dependence at inference time.

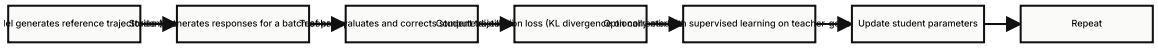
USED BY

THINKING MACHINES | FRONTIER REASONING MODEL DEVELOPERS

CORE IDEA

Standard distillation has the teacher generate responses that the student imitates — but the student never sees its own errors. On-policy distillation closes this gap: the student generates a response (often with chain-of-thought), the teacher reviews and corrects that trajectory, and the student learns from the delta. This reduces exposure bias because the student learns to recover from its own mistakes, not just mimic perfect teacher outputs.

TRAINING PIPELINE



ADVANTAGES

- Better generalization than off-policy distillation
- Reduces exposure bias — student learns from its own mistakes
- Cheaper inference than using teacher at test time
- Student can surpass teacher on specific domains with enough iterations

DISADVANTAGES

- Requires teacher inference during training (compute cost)
- More complicated training pipeline than static distillation
- Teacher corrections must be high quality — noisy corrections hurt
- Risk of student overfitting to teacher correction patterns

BEST FOR

REASONING TASKS WITH VERIFIABLE INTERMEDIATE STEPS

CODING AND AGENTIC TASKS

COMPRESSING LARGE MODELS INTO DEPLOYABLE SIZES

VARIANTS

- Iterative on-policy distillation (student becomes teacher for next round)
- Multi-teacher distillation with weighted corrections

KEY PAPERS

[1] On-Policy Distillation: Learning to Recover from Mistakes — Thinking Machines Research, 2025

[2] Distilling System 2 into System 1 — OpenAI, 2025

[3] STILL-ALIVE: Self-Improvement Through Iterative Learning — Various, 2025

OPEN SOURCE

`LLM Distributed (https://github.com/EleutherAI/llm-distributed)`

`OpenRLHF (https://github.com/OpenRLHF/OpenRLHF)`

COMMON MISTAKES

- *Using a teacher that is not sufficiently more capable than the student*
- *Not periodically refreshing the teacher corrections (student improves, old corrections become stale)*
- *Applying on-policy distillation too early, before the student has basic competence*

FUTURE DIRECTIONS

Online distillation with continuous teacher improvement — the teacher also trains during the process, creating a co-evolving system.

RECIPE 04

Reinforcement Learning from Verifiable Rewards

COMPLEXITY

★★☆☆☆

COMPUTE

Low to Medium — 8–64 GPUs for 1–3 days. Much cheaper than methods requiring a reward model.

PURPOSE

Train reasoning models using binary or structured reward signals from verifiable outcomes (correct answer, pass@k, compiler output) instead of learned reward models.

USED BY

DEEPSEEK R1 | OPENAI O-SERIES | GOOGLE GEMINI REASONING

CORE IDEA

RLVR replaces the learned reward model with a deterministic verifier that checks whether a response satisfies a ground-truth outcome. For math, this means checking if the final answer matches. For code, it means running test cases. For agents, it means success/failure on a task. The verifier provides a clean, binary reward signal that eliminates reward hacking and reduces the complexity of the training pipeline. Combined with GRPO or PPO, RLVR enables scalable training for reasoning.

TRAINING PIPELINE

ADVANTAGES

- No reward model needed — eliminates reward hacking
- Clean, unambiguous training signal
- Scales to arbitrary difficulty as long as answers are verifiable
- Enables training on synthetic data with auto-verification

DISADVANTAGES

- Limited to domains with verifiable outcomes
- Hard to verify complex reasoning chains that are correct but reach the answer differently
- Binary rewards provide less learning signal than dense rewards

BEST FOR

MATHEMATICAL REASONING AND THEOREM PROVING

CODE GENERATION (PASS@K COMPILATION + TEST VERIFICATION)

FACTUAL QA WITH STRUCTURED ANSWER FORMATS

FORMAL VERIFICATION TASKS

VARIANTS

- Process-supervised RLVR (reward per reasoning step via verifier)
- Continuous RLVR (partial credit based on edit distance to correct answer)
- RLVR with self-verification (model checks its own answer before submitting)

KEY PAPERS

[1] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning — DeepSeek AI, 2025

[2] Open Reasoner: Evaluating the Progress of Reasoning — Various, 2025

[3] Let's Verify Step by Step — OpenAI, 2023

OPEN SOURCE

`OpenRLHF (https://github.com/OpenRLHF/OpenRLHF)`

`veRL (https://github.com/volcengine/verl)`

`TRL (https://github.com/huggingface/trl)`

COMMON MISTAKES

- *Using format-based parsing that misses valid but differently formatted correct answers*
- *Training too long on a narrow set of verifiable problems (overfitting to the verifier)*
- *Not including enough difficult problems — model learns to answer easy ones and stops improving*

FUTURE DIRECTIONS

Learned verifiers that can handle open-ended tasks while maintaining the reliability of rule-based checks — the best of both worlds.

RECIPE 05

Preference Optimization

COMPLEXITY

★★☆☆☆

COMPUTE

Low — 1–8 GPUs for 1–2 days for a 7B model with a few thousand preference pairs

PURPOSE

Align language model outputs with human preferences using direct preference pairs, eliminating the need for a separate reward model during training.

USED BY

ANTHROPIC CLAUDE | HUGGINGFACE ZEPHYR | INTEL NEURALCHAT

CORE IDEA

Direct Preference Optimization (DPO) reformulates RLHF without a reward model. Given pairs of preferred/rejected responses, DPO directly optimizes the policy to maximize the log-likelihood of preferred responses while penalizing the rejected ones, using a closed-form mapping between reward functions and optimal policies. Variants like KTO use unpaired preferences and IPO uses a squared-error loss for more stable optimization.

TRAINING PIPELINE

ADVANTAGES

- No reward model training needed — simpler pipeline
- More stable than PPO with learned reward models
- Works with static preference datasets (offline)
- Multiple well-tested open-source implementations

DISADVANTAGES

- Sensitive to preference data quality — garbage in, garbage out
- Can reduce diversity if over-optimized on narrow preferences
- Does not explore new behaviors the way online RL does

BEST FOR

FINAL ALIGNMENT STAGE AFTER SUPERVISED FINE-TUNING

STYLE AND TONE CONTROL

SAFETY AND HARMLESSNESS TRAINING

VARIANTS

- KTO (unpaired preferences, prospect theory loss)
- IPO (squared-error loss, more stable)
- ORPO (no reference model, combines SFT + preference in one step)
- SimPO (sequence-level likelihood as implicit reward)

KEY PAPERS

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model — Rafailov et al., 2023

[2] KTO: Model Alignment as Prospect Theoretic Optimization — Ethayarajh et al., 2024

[3] IPO: A General Framework for Preference Optimization — Azar et al., 2023

[4] ORPO: Monolithic Preference Optimization without Reference Model — Hong et al., 2024

OPEN SOURCE

`TRL (https://github.com/huggingface/trl)`

`Axolotl (https://github.com/axolotl-ai-cloud/axolotl)`

`alignment-handbook (https://github.com/huggingface/alignment-handbook)`

COMMON MISTAKES

- *Using the same data for SFT and DPO (model already prefers chosen responses)*
- *Not using a reference model or using one that is too weak*
- *Beta temperature too low — overfitting to preference data*

FUTURE DIRECTIONS

Online DPO where the model generates new responses and preferences are collected iteratively during training, blending the stability of DPO with the exploration of RL.

RECIPE 06

Constitutional AI

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 8–32 GPUs for 3–7 days depending on model size and constitution rounds

PURPOSE

Train language models to self-critique and revise their own outputs according to a written constitution, reducing harmful outputs without extensive human preference labeling.

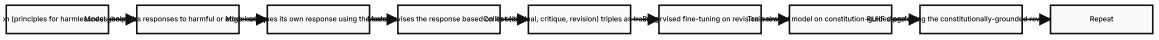
USED BY

ANTHROPIC CLAUDE | SAFETY-ALIGNED OPEN-SOURCE MODELS

CORE IDEA

Constitutional AI replaces much of the human feedback in RLHF with a written set of principles (the "constitution"). The model first generates responses, then critiques its own outputs according to the constitution, and finally revises them. This self-supervision loop produces a dataset of (original → revised) pairs for supervised learning, followed by a standard RLHF stage using a reward model trained on constitution-grounded preferences.

TRAINING PIPELINE



ADVANTAGES

- Reduces reliance on expensive human preference labeling
- Scalable — the constitution can be extended to new principles without new data
- Produces models that can explain their reasoning (via the critique step)
- More consistent than human-labeled preferences

DISADVANTAGES

- Constitution quality is critical — poorly written principles produce poor alignment
- Models can learn to generate safe-sounding but evasive responses
- Requires careful prompt engineering for the critique and revision steps

BEST FOR

SAFETY AND HARMLESSNESS ALIGNMENT

CONTENT MODERATION TRAINING

REDUCING RELIANCE ON HUMAN ANNOTATION

KEY PAPERS

[1] Constitutional AI: Harmlessness from AI Feedback — Bai et al. (Anthropic), 2022

[2] Self-Critiquing Models for Assisting Human Evaluators — Anthropic, 2023

OPEN SOURCE

`TRL Constitutional AI (https://github.com/huggingface/trl)`
`Axolotl (https://github.com/axolotl-ai-cloud/axolotl)`

VARIANTS

- RL from AI Feedback (RLAIF) — using an LLM as the preference judge instead of a constitution
- Self-Play Constitutional AI — iterative improvement of both the model and constitution

COMMON MISTAKES

- *Writing a constitution that is too vague or contradictory*
- *Not iterating on the constitution based on observed failure modes*
- *Using the same constitution for critique and revision without temperature variation*

FUTURE DIRECTIONS

Dynamically updated constitutions that evolve based on deployment feedback, and multi-language constitutions that account for cultural differences in safety preferences.

RECIPE 07

Process Supervision

COMPLEXITY

★★★★☆

COMPUTE

Medium-High — 16–64 GPUs for 4–10 days including PRM training and policy RL

PURPOSE

Provide fine-grained reward signals at each reasoning step rather than only at the final answer, improving training signal density and reducing reward hacking in multi-step tasks.

USED BY

OPENAI O-SERIES | MATH-SHEPHERD | STEP-LEVEL VERIFIERS

CORE IDEA

Instead of rewarding only the final outcome (outcome supervision), process supervision assigns a reward to each intermediate reasoning step. A process reward model (PRM) is trained to evaluate whether each step is correct given the preceding context. This provides a dense training signal that helps the policy learn correct intermediate reasoning, even when the final answer is wrong (and vice versa).

TRAINING PIPELINE

ADVANTAGES

- Denser reward signal improves learning efficiency
- Reduces reward hacking — hard to game 10 step-level rewards
- Enables test-time search (PRM-guided beam search over reasoning paths)
- Better generalization to out-of-distribution problems

DISADVANTAGES

- Step-level annotation is expensive (human or strong model)
- PRM training is more complex than outcome reward modeling
- Defining "steps" is task-dependent and not always natural

BEST FOR

MATHEMATICAL REASONING (MULTI-STEP PROOFS)

CODE GENERATION (STEP-BY-STEP DEBUGGING)

SCIENTIFIC REASONING CHAINS

VARIANTS

- Automatic process supervision (using a strong model as the step annotator instead of humans)
- Automatic curriculum process supervision (adaptive step granularity based on difficulty)

KEY PAPERS

[1] Let's Verify Step by Step — OpenAI, 2023

[2] Math-Shepherd: Verify and Reinforce LLMs Step-by-Step — Wang et al., 2023

[3] Process Reward Model for Mathematical Reasoning — Luo et al., 2024

OPEN SOURCE

`Math-Shepherd (https://github.com/RLHFlow/Math-Shepherd)`
`OpenRLHF (https://github.com/OpenRLHF/OpenRLHF)`

COMMON MISTAKES

- *Using a PRM trained on one domain for another domain without adaptation*
- *Defining steps too coarsely (no benefit over outcome supervision) or too finely (annotation noise)*
- *Not calibrating PRM scores — they can be overconfident on incorrect steps*

FUTURE DIRECTIONS

Unsupervised process supervision where the model self-consistency across multiple rollouts serves as the process reward signal, eliminating the need for annotated step data.

RECIPE 08

Recursive Self-Improvement

COMPLEXITY

★★★☆☆

COMPUTE

Medium — each iteration costs roughly the same as one SFT run; 5-10 iterations typical

PURPOSE

Enable a model to generate its own training data, filter it, and retrain on it in iterative cycles, bootstrapping capability without external supervision.

USED BY

DEEPSEEK R1-ZERO | SELF-REWARDING MODELS | STAR

CORE IDEA

In recursive self-improvement, the model generates candidate solutions or reasoning traces, filters them using a reward signal or self-consistency check, and retrains on the filtered outputs. Each iteration produces a slightly better model, which generates better data for the next round. Key variants include STaR (bootstrapped reasoning), Self-Rewarding (model judges its own outputs), and ReST (iterative self-training with rejection sampling).

TRAINING PIPELINE

ADVANTAGES

- Scales with compute rather than human data collection
- Can discover novel solution strategies not in the training data
- Theoretical path to superhuman performance on narrow domains

DISADVANTAGES

- Risk of mode collapse — model reinforces its own biases
- Quality of the filter/reward is the bottleneck
- Diminishing returns after several iterations without diversity injection

BEST FOR

REASONING TASKS WITH VERIFIABLE OUTCOMES

CODE GENERATION (TEST-BASED FILTERING)

NARROW DOMAINS WITH CLEAR SUCCESS CRITERIA

VARIANTS

- STaR (rationale generation + answer filtering)
- Self-Rewarding (model generates and judges its own outputs)
- ReST (expectation-maximization style self-training)
- Self-Play (model competes against previous versions)

KEY PAPERS

[1] STaR: Bootstrapping Reasoning With Reasoning — Zelikman et al., 2022

[2] Self-Rewarding Language Models — Yuan et al., 2024

[3] ReST: Reinforcement from Self-Training — Gulcehre et al., 2023

[4] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL — DeepSeek AI, 2025

OPEN SOURCE

`STaR (https://github.com/ezelikman/STaR)`
`TRL (https://github.com/huggingface/trl)`
`veRL (https://github.com/volcengine/verl)`

COMMON MISTAKES

- *Filter too loose — model learns from its own errors and degrades*
- *Not enough diversity in prompts — model overfits to narrow distribution*
- *Stopping too early — improvements compound across iterations*

FUTURE DIRECTIONS

Continuous self-improvement during deployment where the model trains on real user interactions filtered by verified outcomes, creating a perpetual improvement loop.

RECIPE 09

Synthetic Curriculum

COMPLEXITY

★★★☆☆

COMPUTE

Medium — requires multiple training rounds; 8–32 GPUs for 3–7 days total

PURPOSE

Generate training data with progressive difficulty, enabling models to learn complex capabilities by starting with easy examples and gradually increasing challenge.

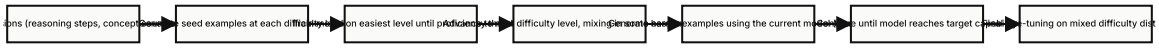
USED BY

PHI SERIES (MICROSOFT) | CODE GENERATION MODELS | MATH LMS

CORE IDEA

Instead of training on a static dataset, synthetic curriculum dynamically generates examples at increasing difficulty levels. Early examples teach basic patterns (format, simple reasoning), while later examples require composition of multiple skills. The difficulty can be controlled by prompt complexity, required reasoning steps, or the number of concepts that must be combined. This mirrors how human learning progresses from simple to complex.

TRAINING PIPELINE



ADVANTAGES

- Faster convergence than uniform difficulty training
- Reduces early overfitting on hard examples the model cannot solve
- Produces more robust models that generalize across difficulty levels
- Works particularly well for code and math

DISADVANTAGES

- Requires reliable difficulty estimation for each example
- Curriculum pacing is a sensitive hyperparameter
- Difficult to design curricula for open-ended tasks

BEST FOR

MATHEMATICAL REASONING (HIERARCHICAL PROBLEM DIFFICULTY)

CODE GENERATION (SYNTAX → SINGLE FUNCTION → MULTI-FILE PROJECTS)

READING COMPREHENSION (SHORT PASSAGES → LONG DOCUMENTS)

VARIANTS

- Automatic curriculum learning (model's loss determines when to advance)
- Self-paced learning (model selects its own difficulty level)
- Transfer curriculum (learn simple skills first, then compose)

KEY PAPERS

[1] Textbooks Are All You Need — Gunasekar et al. (Microsoft), 2023

[2] Phi-2: The Surprising Power of Small Language Models — Microsoft Research, 2024

[3] Curriculum Learning for Language Modeling — Various, 2023

OPEN SOURCE

`Axolotl (https://github.com/axolotl-ai-cloud/axolotl)`
`LMFlow (https://github.com/OptimalScale/LMFlow)`

COMMON MISTAKES

- *Advancing difficulty too fast — model plateaus or regresses*
- *Not maintaining enough easy examples in the mix (catastrophic forgetting)*
- *Using a single difficulty metric when multiple skills are needed*

FUTURE DIRECTIONS

Online curriculum adaptation where the difficulty is adjusted in real-time based on the model's current performance on a validation set, creating an optimal learning trajectory.

RECIPE 10

Interaction Modeling

COMPLEXITY

★★★★☆

COMPUTE

Medium-High — 16–64 GPUs for 5–14 days including data generation and multi-turn training

PURPOSE

Train models to maintain coherent multi-turn interactions, follow complex instructions across conversation turns, and exhibit consistent persona or role behavior.

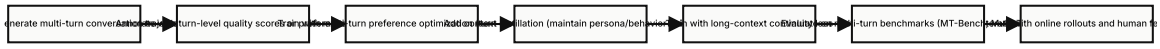
USED BY

CHATGPT | CLAUDE | GEMINI | ROLE-PLAYING MODELS

CORE IDEA

Interaction modeling moves beyond single-turn instruction following to train on full conversation trajectories. The model learns to maintain context across turns, follow evolving instructions, recover from its own mistakes, and exhibit consistent behavior. Training data includes multi-turn conversations with turn-level rewards or preference pairs, often collected from human-human interactions or synthetic rollouts. Key techniques include multi-turn DPO and trajectory-level preference optimization.

TRAINING PIPELINE



ADVANTAGES

- Produces models that feel more natural and coherent in conversation
- Reduces "personality drift" across long conversations
- Enables complex multi-step task completion (booking, planning, research)

DISADVANTAGES

- Multi-turn data is expensive to collect and annotate
- Training is more computationally expensive than single-turn SFT
- Evaluation is more complex — need to assess coherence, not just correctness

BEST FOR

CHAT AND ASSISTANT MODELS

CUSTOMER SERVICE AND SUPPORT BOTS

ROLE-PLAYING AND CHARACTER-BASED INTERACTIONS

LONG-FORM RESEARCH AND ANALYSIS TASKS

KEY PAPERS

[1] Training Language Models with Multi-Turn Preferences — Various, 2024

[2] LongCoT: Long Context Training for Reasoning — Various, 2025

[3] Character-LLM: A Trainable Agent for Role-Playing — Wu et al., 2023

VARIANTS

- Multi-turn DPO (preferences at the trajectory level)
- Context distillation (persona/lore injected as pre-context during training)
- Recursive conversation training (model generates both sides of the conversation)

OPEN SOURCE

`TRL (https://github.com/huggingface/trl)`
`FastChat (https://github.com/lm-sys/FastChat)`

COMMON MISTAKES

- *Training on single turns and expecting multi-turn behavior to emerge*
- *Not varying conversation length in training (model overfits to fixed turn count)*
- *Ignoring the instruction-evolution problem (user instructions change across turns)*

FUTURE DIRECTIONS

Infinite-context interaction models that can maintain coherence across arbitrarily long sessions using memory-augmented architectures and hierarchical conversation representations.

RECIPE 11

State-Space Model Training

COMPLEXITY

★★★★☆

COMPUTE

Medium — similar to Transformer training for equivalent model size; 8–64 GPUs for 5–14 days

PURPOSE

Train linear-complexity sequence models using state-space architectures (Mamba, S4, S5) that match Transformer quality with sub-quadratic scaling to long sequences.

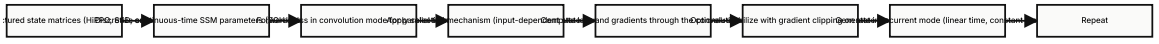
USED BY

MAMBA | JAMBA (AI21) | CADENCE (NVIDIA) | STRIPEDHYENA | NEMOTRON-SS (NVIDIA)

CORE IDEA

State-space models (SSMs) replace attention with a structured linear recurrence that can be computed efficiently as a convolution (for training) or recurrence (for generation). The Mamba architecture introduces selective state-spaces that allow the model to focus on relevant context while maintaining O(n) complexity. Training requires careful initialization of the state matrices and stabilization of the recurrence. The convolution mode enables parallel training across sequence length, while the recurrent mode enables efficient generation.

TRAINING PIPELINE



ADVANTAGES

- Linear scaling with sequence length (vs quadratic for attention)
- Constant memory during generation
- Comparable quality to Transformers on language modeling
- Superior on very long sequences (>16K tokens)

DISADVANTAGES

- Slower at short sequence lengths (convolution overhead)
- Less hardware-efficient than optimized attention (FlashAttention)
- Fewer production-ready implementations than Transformers
- More sensitive to initialization than Transformers

BEST FOR

LONG-DOCUMENT MODELING (LEGAL, MEDICAL, CODE REPOSITORIES)

REAL-TIME GENERATION (SPEECH, STREAMING)

MOBILE AND EDGE DEPLOYMENT (SMALL MEMORY FOOTPRINT)

VARIANTS

- Hybrid SSM-Attention (Mamba-2, Jamba — interleaved SSM and attention layers)
- Multi-scan SSM (bidirectional for encoder tasks)
- Gated SSM (additional gating mechanisms for expressivity)

KEY PAPERS

[1] Mamba: Linear-Time Sequence Modeling with Selective State Spaces — Gu & Dao, 2023

[2] Efficiently Modeling Long Sequences with Structured State Spaces — Gu et al., 2021

[3] Jamba: A Hybrid Transformer-Mamba Language Model — AI21 Labs, 2024

[4] Nemotron-4 340B Technical Report — NVIDIA Research, 2024

OPEN SOURCE

`Mamba (https://github.com/state-spaces/mamba)`
`Cadence (https://github.com/NVIDIA/cadence)`

COMMON MISTAKES

- *Poor initialization causing training instability in early steps*
- *Using low precision (FP16) without loss scaling on state matrix gradients*
- *Not tuning the discretization step size for the specific task*

FUTURE DIRECTIONS

Megascale SSM training with expert parallelism and mixture-of-experts routing, enabling trillion-parameter state-space models that maintain linear complexity.

RECIPE 12

Linear Attention and RWKV Training

COMPLEXITY

★★★☆☆

COMPUTE

Low-Medium — 4–16 GPUs for 2–7 days. Benefits from lower memory for long sequences.

PURPOSE

Train linear-complexity alternatives to softmax attention using kernelized attention (linear transformers) or recurrent formulations (RWKV), enabling efficient long-sequence modeling.

USED BY

RWKV (EAGLE)", "LINEAR TRANSFORMER", "PERFORMER", "REFORMER

CORE IDEA

Linear attention replaces the softmax similarity matrix with a kernelized dot product that can be computed in linear time by changing the order of matrix multiplications. RWKV takes a different approach, combining RNN-style recurrence with transformer-style parallelization through a time-mixing and channel-mixing architecture that is linear in sequence length while remaining parallelizable during training. Both approaches trade some expressivity for computational efficiency.

TRAINING PIPELINE

ADVANTAGES

- O(n) training and O(1) inference memory
- Runs efficiently on CPU for long sequences
- RWKV provides deterministic generation (no sampling randomness)
- Good for resource-constrained deployment

DISADVANTAGES

- Lower quality than Transformers on complex reasoning tasks
- Kernel-based methods lose sparsity benefits of softmax
- RWKV has less parallelization potential than Transformer during training
- Smaller community and ecosystem than Transformer models

BEST FOR

RESOURCE-CONSTRAINED OR EDGE DEPLOYMENT

LONG-DOCUMENT PROCESSING

REAL-TIME STREAMING APPLICATIONS

SCENARIOS REQUIRING DETERMINISTIC OUTPUTS

KEY PAPERS

[1] RWKV: Reinventing RNNs for the Transformer Era — Peng et al., 2023

[2] Eagle RWKV: Faster and Better RWKV — Various, 2025

[3] Efficient Attention: Attention with Linear Complexities — Shen et al., 2018

[4] Rethinking Attention with Performers — Choromanski et al., 2020

OPEN SOURCE

RWKV (https://github.com/BlinkDL/RWKV-LM)

Eagle RWKV (https://github.com/RWKV/EagleRWKV)

VARIANTS

- RWKV-5/Eagle (improved architecture with better parallelism)
- Linear Attention with learned feature maps (data-dependent φ)
- Hybrid linear + softmax attention (linear for long, quadratic for short)

COMMON MISTAKES

- *Using linear attention for short sequences where Transformer is faster*
- *Not tuning the feature map dimension for kernel methods*
- *Overlooking RWKV-specific learning rate schedules (needs warmup + decay)*

FUTURE DIRECTIONS

Training hundred-billion parameter RWKV or linear transformer models at scale, competitive with dense transformers while using a fraction of the inference compute.

PART 2

Vision

7 recipes covering the shift from diffusion-based generation to flow matching and rectified flow, with preference optimization for visual quality, reward-guided fine-tuning, and self-training loops for continuous improvement.

RECIPE 01 PART VISION

Diffusion Preference Optimization

COMPLEXITY

★★★★☆

COMPUTE

High — 8–64 GPUs for 3–10 days depending on model size and preference dataset size

PURPOSE

Align diffusion model outputs with human aesthetic and quality preferences by optimizing the denoising trajectory toward preferred image characteristics.

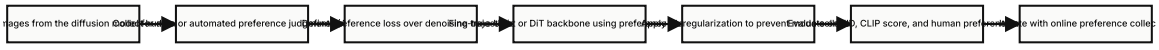
USED BY

STABLE DIFFUSION 3 | MIDJOURNEY | DALL-E 3 ALIGNMENT

CORE IDEA

Diffusion Preference Optimization (DPO for diffusion) extends the preference optimization concept to the denoising process. Given pairs of images where one is preferred (better aesthetics, better prompt alignment), DPO fine-tunes the diffusion model to increase the likelihood of the preferred denoising trajectory. This is done by treating the entire reverse diffusion chain as a multi-step decision process and optimizing the implicit reward defined by the preference pair. Key variants include Diffusion-DPO, SPIN-Diffusion, and DRaFT.

TRAINING PIPELINE



ADVANTAGES

- Directly optimizes for human preferences, not just FID
- Reduces prompt-misalignment issues
- Can correct specific failure modes (anatomy, text rendering)

DISADVANTAGES

- Requires high-quality preference data, ideally human-labeled
- Over-optimization can reduce diversity
- Compute-intensive — each training example requires full denoising

BEST FOR

AESTHETIC FINE-TUNING OF TEXT-TO-IMAGE MODELS

REDUCING ARTIFACT RATES (EXTRA FINGERS, GARBLED TEXT)

BRAND- OR STYLE-SPECIFIC ALIGNMENT

VARIANTS

- SPIN-Diffusion (self-play with preference self-training)
- Direct Reward Fine-Tuning (end-to-end reward optimization)

KEY PAPERS

[1] Diffusion-DPO: Aligning Diffusion Models with Human Preferences — Wallace et al., 2023

[2] DRaFT: Differentiable Rendering for Fine-Tuning Diffusion Models — Clark et al., 2024

[3] Aligning Text-to-Image Models with Human Preference — Xu et al., 2023

OPEN SOURCE

`Diffusion-DPO (https://github.com/baaivision/diffusion-dpo)`

`PickScore (https://github.com/yuvalala1/pickscore)`

COMMON MISTAKES

- *Using a reward model that disagrees with human judgment*
- *Applying too strong a preference weight — reduces diversity to a single style*
- *Not balancing preference data across prompt types*

FUTURE DIRECTIONS

Multi-dimensional preference optimization where different dimensions (aesthetics, safety, style fidelity) are optimized independently and combined at inference time.

RECIPE 02

Flow Matching

COMPLEXITY

★★★☆☆

COMPUTE

Medium-High — similar to diffusion training; 16–128 GPUs for 5–14 days for full-scale models

PURPOSE

Generate samples by learning a continuous normalizing flow between a noise distribution and the data distribution, providing a simpler and more stable alternative to score-based diffusion.

USED BY

STABLE DIFFUSION 3 | FLUX (BLACK FOREST LABS) | SORA | RECTIFIED FLOW MODELS

CORE IDEA

Flow matching replaces the diffusion SDE/ODE with a direct regression objective: given a linear interpolation between noise x_0 and data x_1, the model learns to predict the velocity field dx/dt that maps one to the other. Unlike score matching, flow matching has no time-dependent normalization constants and does not require solving a reverse SDE at inference. The result is simpler training, faster sampling, and better likelihood estimation.

TRAINING PIPELINE

ADVANTAGES

- Simpler training objective than score-matching — no score-normalization
- Deterministic ODE sampling (no SDE discretization error)
- Straight trajectories enable fewer sampling steps (2-10 steps)
- Better likelihood estimation than diffusion models

DISADVANTAGES

- Requires an ODE solver at inference (though efficient ones exist)
- Reflow (trajectory straightening) adds complexity
- Less studied than diffusion for conditional generation tasks

BEST FOR

TEXT-TO-IMAGE GENERATION (STABLE DIFFUSION 3, FLUX)

TEXT-TO-VIDEO (SORA-SCALE FLOW MATCHING)

HIGH-SPEED SAMPLING APPLICATIONS

VARIANTS

- Rectified Flow (reflow step to create straight trajectories)
- Conditional Flow Matching (CFM — conditions on labels or text)
- Discrete Flow Matching (for discrete data like text/tokens)

KEY PAPERS

[1] Flow Matching for Generative Modeling — Lipman et al., 2022

[2] Stable Diffusion 3: Scaling Rectified Flow Transformers — Stability AI, 2024

[3] Flow Matching: Simplified and Generalized — Tong et al., 2024

OPEN SOURCE

`torchcfm (https://github.com/atong01/conditional-flow-matching)`
`Diffusers (https://github.com/huggingface/diffusers)`

COMMON MISTAKES

- *Using too few sampling steps at inference (quality degrades with very coarse solvers)*
- *Not tuning the noise distribution variance*
- *Skipping the reflow step for high-quality generation*

FUTURE DIRECTIONS

Unified flow matching frameworks that handle images, video, 3D, and audio with the same architecture and training procedure — a single generative model for all modalities.

RECIPE 03

Rectified Flow

COMPLEXITY

★★★☆☆

COMPUTE

Medium — reflow costs roughly 0.5x to 1x the original training run; 8–64 GPUs for 2–7 days per round

PURPOSE

Straighten the probability flow ODE trajectories through a reflow procedure, enabling high-quality generation in very few (2-10) sampling steps.

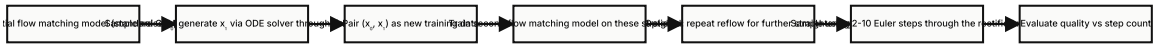
USED BY

STABLE DIFFUSION 3 | INSTAFLOW | CONSISTENCY TRAJECTORY MODELS

CORE IDEA

Rectified flow is a two-step process. First, train a flow matching model on the standard linear interpolation paths. Second, use the learned model to generate new data-noise pairs and train a new model to match the straighter trajectories implied by the first model. This "reflow" procedure straightens the ODE paths, allowing coarse numerical solvers (2-10 steps) to produce high-quality samples. Multiple reflow rounds can be applied.

TRAINING PIPELINE



ADVANTAGES

- 2-10 step sampling with quality approaching 50-step diffusion
- Simple idea that works across modalities
- Each reflow round progressively straightens paths
- Compatible with any flow matching backbone

DISADVANTAGES

- Requires training the model multiple times (reflow rounds)
- Reflow can introduce slight quality degradation if overdone
- Initial model must be trained before reflow can begin

BEST FOR

FAST SAMPLING APPLICATIONS (REAL-TIME GENERATION)

LATENCY-SENSITIVE DEPLOYMENT SCENARIOS

MOBILE AND EDGE IMAGE GENERATION

VARIANTS

- Recursive Rectified Flow (chain multiple reflows)
- Latent Rectified Flow (reflow in the latent space of an autoencoder)

KEY PAPERS

[1] Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow — Liu et al., 2022

[2] InstaFlow: One Step is Enough for High-Quality Diffusion — Liu et al., 2023

OPEN SOURCE

`Rectified Flow (https://github.com/gnobitab/Flow-Matching)`
`Diffusers (https://github.com/huggingface/diffusers)`

COMMON MISTAKES

- *Applying too many reflow rounds (diminishing returns after 2-3)*
- *Not using enough ODE steps in the reflow data generation*
- *Reflowing with a model that is not yet converged*

FUTURE DIRECTIONS

Single-step rectified flow through progressive distillation of the reflowed model, achieving GAN-level speed with diffusion-level quality.

RECIPE 04

Consistency Models

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 8–32 GPUs for 2–5 days for distillation; longer for training from scratch

PURPOSE

Train a model to directly map noise to data in a single step through consistency distillation, enabling one-step generation with quality approaching multi-step diffusion.

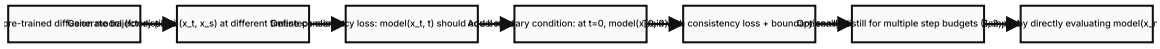
USED BY

OPENAI CONSISTENCY MODELS | LATENT CONSISTENCY MODELS (LCM)

CORE IDEA

Consistency models enforce that the model produces the same output for any point along a diffusion trajectory — the "consistency" property. By learning this mapping, the model can jump from pure noise directly to the data distribution in a single step. Training uses either consistency distillation (from a pre-trained diffusion model) or consistency training (from scratch). Latent Consistency Models apply this in the latent space of an autoencoder for efficient text-to-image generation.

TRAINING PIPELINE



ADVANTAGES

- One-step generation (fastest possible sampling)
- Quality much better than one-step GANs or VAE
- Flexible step budget — can use more steps for higher quality

DISADVANTAGES

- Quality gap vs multi-step diffusion on complex scenes
- Consistency training from scratch is unstable
- Struggles with diverse, high-detail generation

BEST FOR

REAL-TIME IMAGE GENERATION | INTERACTIVE CREATIVE TOOLS | MOBILE AND EDGE DEPLOYMENT | VIDEO GENERATION (WHERE PER-FRAME SPEED MATTERS)

KEY PAPERS

[1] Consistency Models — Song et al. (OpenAI), 2023

[2] Latent Consistency Models: Synthesizing High-Resolution Images — Luo et al., 2023

OPEN SOURCE

`LCM-LoRA (https://github.com/luosiallen/latent-consistency-model)`

`Diffusers LCM (https://github.com/huggingface/diffusers)`

VARIANTS

- Latent Consistency Models (distill in VAE latent space)
- LCM-LoRA (lightweight adaptation for any Stable Diffusion checkpoint)
- Consistency Trajectory Models (multi-step budget flexibility)

COMMON MISTAKES

- *Pushing to single step when 2-4 steps give much better quality*
- *Not tuning the consistency weighting function properly*
- *Using a poor teacher model for distillation*

FUTURE DIRECTIONS

Progressive consistency: a single model that smoothly trades off speed and quality at inference time, from 1 step to 50 steps, without re-training.

RECIPE 05

Vision RL

COMPLEXITY

★★★★☆

COMPUTE

Medium — 8–32 GPUs for 2–7 days depending on reward model and generator size

PURPOSE

Apply reinforcement learning to vision tasks by using differentiable reward functions (aesthetic scoring, CLIP alignment, or human feedback) to fine-tune generative vision models.

USED BY

DALL-E RL FINE-TUNING | IMAGEN REWARD TRAINING | AESTHETIC FINE-TUNING PIPELINES

CORE IDEA

Vision RL treats the image generation process as a policy that produces visual outputs, which are then scored by a reward function. The reward can be a learned aesthetic scorer, a CLIP-based alignment score, or a human preference model. By backpropagating through the reward function, the vision model is fine-tuned to produce higher-reward outputs. Key challenges include making non-differentiable rewards differentiable (via score function estimators or Gumbel-softmax) and preventing reward overfitting.

TRAINING PIPELINE

ADVANTAGES

- Can optimize for arbitrary reward functions
- Directly improves human-judged quality
- Compatible with any generative vision architecture

DISADVANTAGES

- Policy gradient methods have high variance
- Reward models can be hacked (over-optimization)
- Requires careful balancing of reward vs diversity

BEST FOR

IMPROVING IMAGE AESTHETICS

FINE-TUNING FOR BRAND/STYLE CONSISTENCY

OPTIMIZING FOR SPECIFIC METRICS (CLIP SCORE, FID)

VARIANTS

- Reward-guided diffusion (guidance-scale style reward conditioning)
- Reinforcement fine-tuning with human feedback for vision

KEY PAPERS

[1] Fine-Tuning Image Generators with Human Preferences — Lee et al., 2023

[2] Rewarding Progress: Scaling Reward Models for Vision — Various, 2024

OPEN SOURCE

`DRaFT (https://github.com/kvablack/draft)`

`Diffusers (https://github.com/huggingface/diffusers)`

COMMON MISTAKES

- *Over-optimizing the reward at the cost of diversity*
- *Using a reward model that is not calibrated for the target domain*
- *Applying too strong KL regularization (no improvement)*

FUTURE DIRECTIONS

Multi-objective vision RL where diverse reward functions (aesthetics, safety, brand consistency, text alignment) are optimized simultaneously using preference-based multi-objective RL.

RECIPE 06

Image Reward Models

COMPLEXITY

★★☆☆☆

COMPUTE

Low — 1–8 GPUs for 1–3 days for typical model sizes (CLIP-L or ViT-L backbone)

PURPOSE

Train scoring models that evaluate image quality, aesthetics, and prompt alignment, enabling automated evaluation and reward signals for vision model fine-tuning.

USED BY

PICKSCORE | IMAGEREWARD | HPS (HUMAN PREFERENCE SCORE) V2

CORE IDEA

Image reward models are trained on human preference judgments (image A vs B given a prompt) to predict which image a human would prefer. They typically use a vision-language backbone (CLIP, BLIP) with a lightweight scoring head. The reward model outputs a scalar score for any image-prompt pair, serving as a proxy for human judgment. These scores are used for model evaluation, prompt engineering, and as reward signals in RL-based fine-tuning.

TRAINING PIPELINE

ADVANTAGES

- Enables automated evaluation at scale
- Provides a smooth reward signal for RL fine-tuning
- Can predict multiple quality dimensions

DISADVANTAGES

- Bias inherits from the preference data collector
- Generalizes poorly to out-of-distribution prompts/styles
- Can be gamed by adversarial image features

BEST FOR

AUTOMATED EVALUATION OF GENERATED IMAGES

REWARD SIGNAL FOR DIFFUSION RL FINE-TUNING

FILTERING AND RANKING IMAGE GENERATION OUTPUTS

VARIANTS

- Multi-aspect reward models (separate heads for aesthetics, alignment, artifacts)
- Video reward models (temporal consistency scoring)

KEY PAPERS

[1] ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation — Xu et al., 2023

[2] PickScore: Human Preference Scoring for Text-to-Image Generation — Kirstain et al., 2024

[3] Human Preference Score v2: A Better Benchmark for Image Generation — Wu et al., 2023

OPEN SOURCE

`ImageReward (https://github.com/THUDM/ImageReward)`

`PickScore (https://github.com/yuvalala1/pickscore)`

`HPS v2 (https://github.com/tgxs002/HPSv2)`

COMMON MISTAKES

- *Training on too few prompt-image pairs (underfitting preferences)*
- *Using the reward model on prompts very different from training distribution*
- *Not normalizing scores across different prompt types*

FUTURE DIRECTIONS

Reward models that produce fine-grained pixel-level feedback, identifying exactly which regions of an image are problematic, enabling targeted improvement.

RECIPE 07

Self-Training for Vision

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 2–3x the cost of a single supervised training run; 4–32 GPUs for 3–10 days

PURPOSE

Improve vision models iteratively by generating pseudo-labels or synthetic training examples and retraining on the augmented dataset, reducing reliance on human annotation.

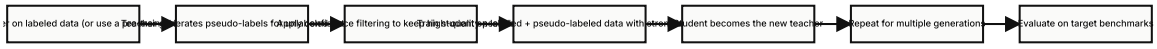
USED BY

NOISY STUDENT (EFFICIENTNET) | SELF-TRAINING FOR DETECTION | DINO SELF-DISTILLATION

CORE IDEA

Self-training in vision follows a teacher-student loop: a teacher model generates pseudo-labels on unlabeled data, and a student is trained on the combined labeled + pseudo-labeled data. The student then becomes the teacher for the next iteration. This can be combined with data augmentation, noise injection, and consistency regularization. Modern approaches like DINO and DINOv2 use self-distillation with careful augmentation strategies to learn visual features without any labels.

TRAINING PIPELINE



ADVANTAGES

- Leverages large amounts of unlabeled or synthetic data
- Iterative improvement without new human labels
- Works across classification, detection, and segmentation

DISADVANTAGES

- Pseudo-label noise amplifies across iterations
- Requires careful confidence threshold tuning
- Expensive — requires multiple full training runs

BEST FOR

SEMI-SUPERVISED VISION TASKS WITH ABUNDANT UNLABELED DATA

DOMAIN ADAPTATION (LABELED SOURCE, UNLABELED TARGET)

PRE-TRAINING VISION BACKBONES

KEY PAPERS

[1] Self-Training with Noisy Student Improves ImageNet Classification — Xie et al. (Google), 2019

[2] DINOv2: Learning Robust Visual Features without Supervision — Meta AI, 2023

[3] STAC: Self-Training for Object Detection — Sohn et al., 2020

VARIANTS

- Noisy Student Training (inject noise during student training for robustness)
- DINO (self-distillation with no labels, using CLS token)
- Consistency-based self-training (enforce prediction consistency under augmentations)

OPEN SOURCE

`DINO (https://github.com/facebookresearch/dino)`
`SEER (https://github.com/facebookresearch/vissl)`

COMMON MISTAKES

- *Using a teacher that is too weak (low-quality pseudo-labels)*
- *Not filtering pseudo-labels by confidence*
- *Applying identical augmentations to teacher and student*

FUTURE DIRECTIONS

Vision self-training that incorporates generative models: a diffusion model generates synthetic training images with perfect labels, which are then used to train vision encoders in an endless data flywheel.

PART 3

3D Generation

8 recipes spanning multi-view diffusion, Gaussian splatting supervision, mesh generation, neural field training, scene-level generation, world-state prediction, synthetic 3D pretraining, and animation distillation — one of the fastest-moving areas with few consolidated references.

RECIPE 01 | PART 3D-GENERATION

Multi-View Diffusion

COMPLEXITY

★★★★☆

COMPUTE

High — 16–64 GPUs for 7–21 days for full model training

PURPOSE

Train a diffusion model to generate multiple consistent views of a 3D object from a single input image or text prompt, enabling 3D asset creation without explicit 3D supervision.

USED BY

ZERO-1-TO-3 | MVDREAM | CAT3D | STABLE ZERO123

CORE IDEA

Multi-view diffusion extends standard image diffusion to generate several images of the same object from different camera viewpoints, with cross-view consistency. The model conditions on a reference image and relative camera pose, and generates N views simultaneously using cross-attention between views. Training requires either multi-view renderings of 3D assets or video frames (where consecutive frames approximate multi-view). The generated views can then be fed into a 3D reconstruction method (NeRF, Gaussian Splatting) for full 3D asset creation.

TRAINING PIPELINE

ADVANTAGES

- Produces 3D-consistent multi-view outputs
- Works from a single image or text prompt
- No explicit 3D supervision needed during training
- Compatible with existing 3D reconstruction pipelines

DISADVANTAGES

- Requires large datasets of multi-view renderings
- Cross-view consistency is not always perfect
- Camera pose estimation is an additional engineering challenge

BEST FOR

SINGLE-IMAGE TO 3D ASSET GENERATION

TEXT-TO-3D GENERATION (VIA TEXT-TO-IMAGE MODELS)

VIEW SYNTHESIS FOR 3D RECONSTRUCTION

VARIANTS

- Score Distillation Sampling (SDS) — use pretrained multi-view diffusion as 3D prior
- Video-based multi-view (fine-tune on video to learn view consistency)

KEY PAPERS

[1] Zero-1-to-3: Zero-shot One Image to 3D Object — Liu et al., 2023

[2] MVDream: Multi-View Diffusion for 3D Generation — Shi et al., 2023

[3] CAT3D: Create Anything in 3D — Meta AI, 2024

OPEN SOURCE

`Zero-1-to-3 (https://github.com/cvlab-columbia/zero123)`

`MVDream (https://github.com/bytedance/mvdream)`

`Stable Zero123 (https://github.com/Stability-AI/generative-models)`

COMMON MISTAKES

- *Not enough camera pose diversity in training data*
- *Training without cross-view attention (views are inconsistent)*
- *Using too few views for reconstruction (3 views minimum, 6+ recommended)*

FUTURE DIRECTIONS

Real-time multi-view generation that can produce hundreds of consistent views from a single input, enabling instant 3D reconstruction for AR/VR and gaming applications.

RECIPE 02

Gaussian Splatting Supervision

COMPLEXITY

★★★☆☆

COMPUTE

Low-Medium — 1–4 GPUs for 15 minutes to 2 hours per scene

PURPOSE

Train 3D Gaussian Splatting representations from multi-view images or video, optimizing the position, covariance, color, and opacity of Gaussian primitives to reconstruct a 3D scene.

USED BY

- 3D GAUSSIAN SPLATTING (KERBL ET AL.)
- SPLATAM
- NERFSTUDIO GS PIPELINES

CORE IDEA

3D Gaussian Splatting represents a scene as a collection of anisotropic 3D Gaussians, each defined by a position, covariance matrix, color (with spherical harmonics for view-dependence), and opacity. Training optimizes these parameters through differentiable rendering: Gaussians are projected to the image plane, rasterized, and compared to training views using photometric loss. Adaptive density control splits and clones Gaussians where reconstruction error is high, and prunes near-transparent ones.

TRAINING PIPELINE

ADVANTAGES

- Real-time rendering (>100 FPS at 1080p)
- Fast training (minutes to hours, vs days for NeRF)
- Excellent visual quality with sharp details

DISADVANTAGES

- High memory usage (millions of Gaussians at full quality)
- Requires good initial point cloud (SfM) for best results
- Less compact than NeRF (larger storage footprint)

BEST FOR

- NOVEL VIEW SYNTHESIS FROM MULTI-VIEW PHOTOS
- REAL-TIME 3D SCENE RENDERING
- 3D RECONSTRUCTION FROM VIDEO
- DIGITAL TWINS AND SCENE SCANNING

KEY PAPERS

[1] 3D Gaussian Splatting for Real-Time Radiance Field Rendering — Kerbl et al., 2023

[2] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis — Luiten et al., 2023

[3] DrivingGaussian: Composite Gaussian Splatting for Autonomous Driving — Zhou et al., 2023

VARIANTS

- Dynamic Gaussian Splatting (4D Gaussians for video/scene flow)
- Semantic Gaussian Splatting (add semantic features to each Gaussian)
- Compressed Gaussian Splatting (vector quantization for smaller storage)

OPEN SOURCE

`3D Gaussian Splatting (https://github.com/graphdeco-inria/gaussian-splatting)`

`Nerfstudio GS (https://github.com/nerfstudio-project/gsplat)`

`SplaTAM (https://github.com/spla-tam/SplaTAM)`

COMMON MISTAKES

- *Using too few training views (< 20 leads to poor reconstruction)*
- *Not tuning the adaptive density control thresholds*
- *Neglecting spherical harmonics for view-dependent effects*

FUTURE DIRECTIONS

Feed-forward Gaussian Splatting prediction directly from a single image or video, eliminating per-scene optimization entirely and enabling real-time 3D reconstruction.

RECIPE 03

Mesh Diffusion

COMPLEXITY

★★★★★

COMPUTE

High — 16–64 GPUs for 7–21 days

PURPOSE

Train diffusion models that directly generate 3D mesh structures (vertices, faces, topology) rather than 2D representations, producing ready-to-use 3D assets.

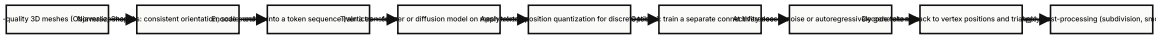
USED BY

MESHGPT | POLYGEN | MESHDIFFUSION | XNOR

CORE IDEA

Mesh diffusion operates directly on 3D mesh representations instead of generating 2D views for reconstruction. The training pipeline involves encoding meshes into a structured representation (vertex sequences, triangle sets, or latent codes), then learning the diffusion process on this representation. MeshGPT uses a transformer to autoregressively generate vertex positions and face connectivity. PolyGen generates meshes by first predicting vertex positions, then predicting the face topology. This produces watertight, production-ready meshes without post-processing.

TRAINING PIPELINE



ADVANTAGES

- Directly produces 3D meshes, not implicit representations
- Quality improves with more training data and compute
- Can generate watertight, manifold meshes

DISADVANTAGES

- Fixed vertex count limits geometric complexity
- Mesh tokenization is lossy (quantization artifacts)
- Requires large datasets of clean 3D meshes
- More computationally expensive than implicit methods

BEST FOR

GAME AND AR/VR ASSET GENERATION

3D MODELING ASSISTANCE AND AUTO-COMPLETION

SHAPE INTERPOLATION AND EDITING

VARIANTS

- Octree-based mesh generation (variable resolution)
- Latent mesh diffusion (diffuse in a compressed latent space)
- Text-conditional mesh generation (via CLIP or similar text encoder)

KEY PAPERS

[1] MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers — Siddiqui et al., 2023

[2] PolyGen: An Autoregressive Generative Model of 3D Meshes — Nash et al. (DeepMind), 2021

[3] MeshDiffusion: Score-Based Generative 3D Mesh Modeling — Liu et al., 2023

OPEN SOURCE

`MeshGPT (https://github.com/lucidrains/meshgpt-pytorch)`

`PolyGen (https://github.com/deepmind/deepmind-research/tree/master/polygen)`

COMMON MISTAKES

- *Using too coarse vertex quantization (visible faceting artifacts)*
- *Training on poorly cleaned mesh data (non-manifold geometry)*
- *Not handling topological degeneracies in training data*

FUTURE DIRECTIONS

Unified mesh generation that simultaneously predicts geometry, topology, texture coordinates, and materials in a single diffusion or autoregressive model, producing game-ready assets directly.

RECIPE 04

Neural Field Training

COMPLEXITY

★★★☆☆

COMPUTE

Low-Medium — 1–4 GPUs for 15 minutes (Instant NGP) to 24 hours (full NeRF)

PURPOSE

Train implicit neural representations (NeRF, Instant NGP, tri-plane) that encode 3D scenes as continuous functions mapping spatial coordinates to density and color, optimized from multi-view images.

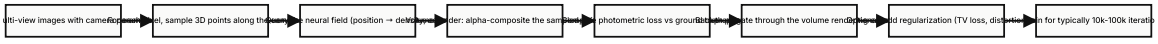
USED BY

NERF | INSTANT NGP | NEURALANGELO | TRI-PLANE (EG3D)

CORE IDEA

Neural fields represent a 3D scene as a neural network that maps a 3D coordinate (and optionally viewing direction) to density and color. Training uses differentiable volume rendering: for each pixel, points along the camera ray are sampled, their density and color are evaluated by the network, and alpha-composited to produce a pixel color. The loss between rendered and ground truth pixel colors drives the optimization. Modern variants use efficient grid-based representations (Instant NGP), tri-plane hybrid representations (EG3D), or hash encoding for fast training.

TRAINING PIPELINE



ADVANTAGES

- Continuous representation (arbitrary resolution)
- Compact storage (a single MLP for an entire scene)
- Handles complex topology and transparent objects

DISADVANTAGES

- Slow rendering (requires many network evaluations per pixel)
- Slow training (hours to days for complex scenes)
- Requires accurate camera parameters

BEST FOR

NOVEL VIEW SYNTHESIS FROM PHOTOS

SCENE CAPTURE AND RECONSTRUCTION

3D-AWARE IMAGE GENERATION (EG3D)

VARIANTS

- Tri-plane / EG3D (hybrid explicit-implicit for 3D-aware GANs)
- Zip-NeRF (multi-scale hash grid for anti-aliasing)
- NeRF in the Wild (handles variable lighting and occluders)

KEY PAPERS

[1] NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis — Mildenhall et al., 2020

[2] Instant Neural Graphics Primitives with a Multiresolution Hash Encoding — Müller et al. (NVIDIA), 2022

[3] Neuralangelo: High-Fidelity Neural Surface Reconstruction — Li et al. (NVIDIA), 2023

OPEN SOURCE

`Nerfstudio (https://github.com/nerfstudio-project/nerfstudio)`

`Instant NGP (https://github.com/NVlabs/instant-ngp)`

`Neuralangelo (https://github.com/NVlabs/neuralangelo)`

COMMON MISTAKES

- *Inaccurate camera poses (biggest source of NeRF failure)*
- *Training with too few views (< 15 for complex scenes)*
- *Not using scene contraction for unbounded scenes*

FUTURE DIRECTIONS

Neural fields that generalize across scenes: a single network pre-trained on thousands of scenes can reconstruct a new scene from just 1-3 views by inverting a small latent code.

RECIPE 05

Scene Graph Planning

COMPLEXITY

★★★★☆

COMPUTE

Medium — 8–32 GPUs for 3–10 days depending on scene complexity

PURPOSE

Generate structured 3D scenes by explicitly modeling object relationships, spatial arrangements, and scene composition through graph-based representations.

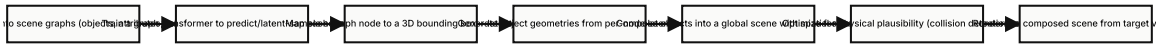
USED BY

SCENEGRAPHNET | 3D-SIS | STRUCTURED 3D GENERATION | GRAPH-TO-3D PIPELINES

CORE IDEA

Scene graph planning decomposes 3D scene generation into a structured process: first predict or plan the scene graph (objects + relationships + spatial layout), then generate the geometry for each object. The scene graph encodes "the cup is ON the table, the table is NEXT TO the chair" as a structured representation. Training involves learning the distribution over valid scene graphs and per-object generators that can be composed into a coherent 3D scene.

TRAINING PIPELINE



ADVANTAGES

- Explicitly controllable scene composition
- Handles complex multi-object scenes
- Supports editing by modifying the scene graph

DISADVANTAGES

- Limited by the scene graph vocabulary
- Per-object generation is computationally expensive
- Global consistency (lighting, shadows) across objects is hard

BEST FOR

INDOOR SCENE GENERATION (ROOMS, FURNITURE LAYOUTS) | INTERACTIVE 3D SCENE EDITING | PROCEDURAL CONTENT GENERATION FOR GAMES

KEY PAPERS

[1] SceneGraphNet: Neural Message Passing for 3D Scene Understanding — Various, 2023

[2] 3D Scene Generation via Scene Graphs — Various, 2023

VARIANTS

- Latent scene graphs (diffusion over graph latents)
- Text-to-scene-graph parsing + 3D generation pipeline

OPEN SOURCE

`SceneGraphNet (https://github.com/scenegraphnet/scenegraphnet)`

COMMON MISTAKES

- *Using too sparse a graph (missing important spatial relations)*
- *Not enforcing physical constraints (objects floating/intersecting)*
- *Training on scenes with inconsistent graph annotations*

FUTURE DIRECTIONS

Dynamic scene graphs that model not just static layouts but object interactions, affordances, and functional relationships — enabling 3D scenes that can be interacted with, not just viewed.

RECIPE 06

World-State Prediction

COMPLEXITY

★★★★★

COMPUTE

High — 16–64 GPUs for 10–30 days for large-scale 3D world models

PURPOSE

Train models to predict future 3D states of a scene from current observations, enabling physics-aware 3D forecasting for robotics, autonomous driving, and simulation.

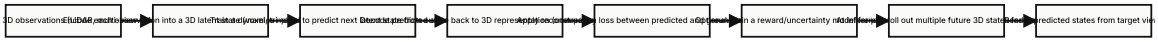
USED BY

WORLD MODELS (3D VARIANTS) | DRIVEWORLD | OCCNET PREDICTION

CORE IDEA

World-state prediction extends next-frame prediction from 2D pixels to full 3D scene representations. Given a sequence of 3D observations (point clouds, voxel grids, or neural fields), the model learns a dynamics model that predicts the next 3D state. This is trained on sequences of real or simulated 3D data with a reconstruction or occupancy loss. The predicted 3D state can be rendered from any viewpoint, enabling the model to "imagine" what will happen next in 3D.

TRAINING PIPELINE



ADVANTAGES

- Enables 3D-aware planning and forecasting
- Multimodal future prediction (multiple possible futures)
- Renders from arbitrary viewpoints for interpretability

DISADVANTAGES

- Expensive — 3D state representation requires more compute than 2D
- Limited by the quality of 3D observations
- Long-horizon prediction remains difficult

BEST FOR

AUTONOMOUS DRIVING SCENE FORECASTING

ROBOTICS PLANNING IN KNOWN ENVIRONMENTS

PHYSICS SIMULATION AND "WHAT-IF" SCENARIO MODELING

VARIANTS

- Latent world models (predict in latent space, decode to 3D only for visualization)
- Diffusion-based world models (diffuse over possible future 3D states)

KEY PAPERS

[1] DriveWorld: 4D Pre-trained Scene Understanding for Autonomous Driving — Various, 2024

[2] Occupancy Prediction for Autonomous Driving — Tesla / Various, 2023

OPEN SOURCE

`DriveWorld (https://github.com/driveworld/driveworld)`

`OccNet (https://github.com/opendrivelab/occnet)`

COMMON MISTAKES

- *Predicting in 2D and projecting to 3D (loses 3D consistency)*
- *Using too simple a dynamics model (linear — fails on complex scenes)*
- *Not modeling uncertainty in the future state prediction*

FUTURE DIRECTIONS

Large-scale 3D world models pre-trained on internet-scale video that can be fine-tuned for any downstream task — the 3D equivalent of large language models.

RECIPE 07

Procedural Supervision

COMPLEXITY

★★★★☆

COMPUTE

Medium-High — rendering pipeline + training; 8–64 GPUs for 5–21 days

PURPOSE

Use procedurally generated 3D data with perfect ground truth labels to supervise 3D vision models, bypassing the need for expensive real-world 3D annotation.

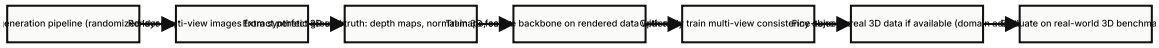
USED BY

OBJAVERSE-XL TRAINING | 3D FOUNDATION MODELS | CLIP FOR 3D

CORE IDEA

Procedural supervision generates synthetic 3D training data using rendering engines or procedural generation, providing perfect labels (depth, normals, segmentation, correspondences) at no human cost. The training pipeline renders large volumes of synthetic 3D scenes with varied geometry, materials, and lighting, then uses the automatically-generated labels to supervise 3D backbone networks. Models pretrained this way transfer surprisingly well to real-world 3D tasks.

TRAINING PIPELINE



ADVANTAGES

- Perfect labels at zero human annotation cost
- Unlimited training data (scale to billions of examples)
- Controllable distribution (balance rare cases)
- Improves real-world performance through domain randomization

DISADVANTAGES

- Sim-to-real gap: synthetic data has visual domain differences
- Procedural generation pipeline is engineering-intensive
- Cannot capture real-world corner cases that were not programmed

BEST FOR

PRE-TRAINING 3D FOUNDATION MODELS

DEPTH AND NORMAL ESTIMATION

MULTI-VIEW CORRESPONDENCE LEARNING

ANY TASK WITH A WELL-DEFINED RENDERING PIPELINE

KEY PAPERS

[1] Objaverse: A Universe of Annotated 3D Objects — Deitke et al., 2022

[2] Objaverse-XL: A Universe of 10M+ 3D Objects — Deitke et al., 2023

[3] 3D Foundation Models: Pre-training for 3D Vision — Various, 2024

OPEN SOURCE

`Objaverse (https://github.com/allenai/objaverse-xl)`

`BlenderProc (https://github.com/DLR-RM/BlenderProc)`

VARIANTS

- Domain randomization (vary rendering parameters to bridge sim-to-real)
- Mixed reality data (composite rendered objects into real backgrounds)

COMMON MISTAKES

- *Not randomizing lighting, textures, and camera positions enough*
- *Using a fixed synthetic distribution that does not match real-world priors*
- *Over-training on synthetic data (overfitting to render artifacts)*

FUTURE DIRECTIONS

Generative procedural supervision where a diffusion model creates infinite diverse training scenes dynamically, adapting the synthetic data distribution to the model's current failure modes.

RECIPE 08

Animation Distillation

COMPLEXITY

★★★★☆

COMPUTE

Medium — 4–16 GPUs for 3–7 days for training; real-time at inference

PURPOSE

Transfer animation rigs and motion sequences from template 3D assets to newly generated 3D objects, enabling automatic animation of generative 3D content.

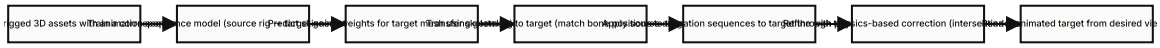
USED BY

ANIMATEDIFF (3D EXTENSION) | MOTION TRANSFER FOR 3D | SKINNING-BASED GENERATION

CORE IDEA

Animation distillation transfers existing animation data (skeletal rigs, blend shapes, skinning weights) to novel 3D geometries. Given a source 3D asset with an animation rig, and a target 3D object in a similar pose, the model learns to predict skinning weights and rig parameters for the target object. This enables generated 3D assets to be automatically animated using existing animation libraries, bypassing the expensive manual rigging process.

TRAINING PIPELINE



ADVANTAGES

- Animates generated 3D content without manual rigging
- Leverages existing high-quality animation libraries
- Compatible with standard game engine pipelines (FBX, glTF)

DISADVANTAGES

- Requires similar topology between source and target
- Degraded quality for very different geometries
- Complex motion sequences may not transfer cleanly

BEST FOR

CHARACTER ANIMATION FROM GENERATED 3D ASSETS

GAME ASSET PIPELINE AUTOMATION

BATCH ANIMATION OF 3D ASSET LIBRARIES

VARIANTS

- Neural skinning (predict vertex transformations directly, no skeleton)
- Example-based animation (interpolate between keyframe poses)

KEY PAPERS

[1] Animation from Motion: Transferring Animations to 3D Assets — Various, 2024

[2] RigNet: Neural Rigging for Articulated Characters — Xu et al., 2020

OPEN SOURCE

`RigNet (https://github.com/zhan-xu/RigNet)`

`Blender auto-rigging addons`

COMMON MISTAKES

- *Using source and target meshes with very different vertex counts*
- *Not handling non-rigid deformations (cloth, soft tissue)*
- *Applying animations designed for one scale to very different asset sizes*

FUTURE DIRECTIONS

Animation generation directly from text or video: describe "a walking cycle for this dragon" and the full rig + animation is generated without any source template.

PART 4

Speech

5 recipes tracing the arc from self-supervised speech tokenization (HuBERT, EnCodec) through codec language models (VALL-E, AudioLM) to speech RL fine-tuning, multi-speaker distillation, and few-shot voice cloning.

RECIPE 01 PART SPEECH

Speech Token Models

COMPLEXITY

★★★★☆

COMPUTE

High — 16–64 GPUs for 5–21 days for large-scale (HuBERT Large, EnCodec)

PURPOSE

Learn discrete or continuous speech representations from raw audio through self-supervised training, providing high-quality tokenization for downstream speech generation and understanding.

USED BY

HUBERT | WAV2VEC 2.0 | ENCODEC | DAC (DESCRIPT AUDIO CODEC)

CORE IDEA

Speech token models convert raw audio waveforms into discrete tokens (for language-model-style processing) or continuous representations (for feature extraction). The training is self-supervised: HuBERT uses a clustering objective where the model predicts masked audio regions; wav2vec 2.0 uses contrastive prediction over latent speech representations; EnCodec and DAC are neural audio codecs trained with reconstruction loss and perceptual losses. The resulting tokens can be used for speech synthesis (as inputs to a codec LM), speech recognition, or emotion/speaker analysis.

TRAINING PIPELINE

ADVANTAGES

- Learns from unlabeled speech data at scale
- Encoder representations are useful for many downstream tasks
- Neural codecs achieve high compression (1.5-12 kbps) with good quality

DISADVANTAGES

- Performance degrades on noisy or accented speech
- High compute cost especially for codec training
- Self-supervised representations can miss phonetic detail

BEST FOR

SPEECH REPRESENTATION LEARNING FOR ASR

NEURAL AUDIO COMPRESSION FOR STREAMING

TOKENIZATION FOR SPEECH LANGUAGE MODELS

VARIANTS

- WavLM (HuBERT with utterance-level training)
- SpeechTokenizer (hierarchical semantic + acoustic tokens)

KEY PAPERS

[1] HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units — Hsu et al. (Meta AI), 2021

[2] wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations — Baevski et al. (Meta AI), 2020

[3] EnCodec: High Fidelity Neural Audio Compression — Défossez et al. (Meta AI), 2022

[4] DAC: Descript Audio Codec — Kumar et al. (Descript), 2024

OPEN SOURCE

`Fairseq (https://github.com/facebookresearch/fairseq)`

`EnCodec (https://github.com/facebookresearch/encodec)`

`DAC (https://github.com/descriptinc/descript-audio-codec)`

COMMON MISTAKES

- *Not using enough audio data (HuBERT needs 10k+ hours)*
- *Using a single quantization level when RVQ with multiple levels is better*
- *Neglecting audio preprocessing (sample rate, normalization)*

FUTURE DIRECTIONS

Universal audio tokenizer that handles speech, music, and environmental sounds with a single model, enabling cross-modal audio generation from text or video inputs.

RECIPE 02

Codec Language Models

COMPLEXITY

★★★★☆

COMPUTE

High — 16–128 GPUs for 7–30 days for production-scale models

PURPOSE

Generate speech by training language models on discrete audio tokens from neural codecs, enabling text-to-speech, voice cloning, and speech-to-speech translation with natural prosody.

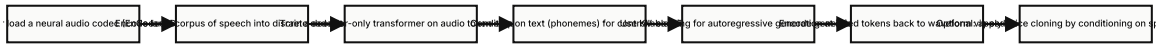
USED BY

VALL-E | AUDIOLM | USM (GOOGLE) | SOUNDSTORM

CORE IDEA

Codec language models treat audio as a language: the speech signal is first encoded into discrete tokens by a neural audio codec (EnCodec, DAC), then a language model (decoder-only transformer) is trained to predict these tokens autoregressively. VALL-E uses a phoneme-conditioned autoregressive model to generate codec tokens from text. AudioLM extends this to generate audio from a short prompt (continuation). SoundStorm adds parallel decoding for speed. The key insight is that language model scaling laws apply to audio tokens just as they do to text.

TRAINING PIPELINE



ADVANTAGES

- Highly natural prosody and expressiveness
- Scales with compute (larger models = better speech)
- Handles multiple tasks in one model (TTS, voice cloning, continuation)

DISADVANTAGES

- Autoregressive generation is slow (needs real-time mitigation)
- Codec artifacts can be audible at low bitrates
- Requires large amounts of speech data (10k+ hours)

BEST FOR

TEXT-TO-SPEECH WITH NATURAL PROSODY

VOICE CLONING FROM SHORT SAMPLES

SPEECH-TO-SPEECH TRANSLATION

MUSIC GENERATION (AUDIOLM, MUSICLM)

KEY PAPERS

[1] VALL-E: Neural Codec Language Models for Text-to-Speech Synthesis — Wang et al. (Microsoft), 2023

[2] AudioLM: A Language Modeling Approach to Audio Generation — Google, 2022

[3] SoundStorm: Efficient Parallel Audio Generation — Google, 2023

OPEN SOURCE

`VALL-E (https://github.com/microsoft/vall-e)`

`AudioLM (https://github.com/google-research/audiolm)`

VARIANTS

- VALL-E-X (cross-lingual voice cloning)
- MusicLM (music generation from text)
- SoundStorm (non-autoregressive parallel decoding)

COMMON MISTAKES

- *Using codec tokens that are too coarse (audible artifacts)*
- *Not phoneme-aligning text and audio for TTS conditioning*
- *Generating with too much sampling temperature (garbled audio)*

FUTURE DIRECTIONS

Multi-stream codec LMs that generate speech, music, and sound effects simultaneously with text and video conditioning — a single "audio language model" for all audio generation.

RECIPE 03

Speech RL

COMPLEXITY

★★★★☆

COMPUTE

Medium-High — 8–32 GPUs for 3–10 days including reward model training

PURPOSE

Apply reinforcement learning to fine-tune speech generation models for objective quality metrics, naturalness, and expressiveness beyond supervised training.

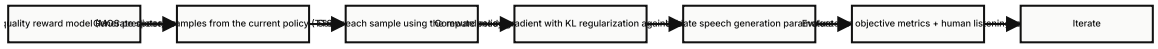
USED BY

NATURALSPEECH 3 RL | VOICEBOX ALIGNMENT | VALL-E RL FINE-TUNING

CORE IDEA

Speech RL applies policy gradient methods to speech generation models, using reward models trained on human judgments of speech quality or objective metrics (MOS prediction, intelligibility scores). The speech generator is treated as a policy that produces audio, which is scored by the reward model. This allows optimization for aspects of speech quality that are hard to capture with supervised loss: natural prosody, expressiveness, listener preference.

TRAINING PIPELINE



ADVANTAGES

- Directly optimizes for human-perceived quality
- Can correct artifacts that supervised losses miss
- Adaptable to specific deployment requirements

DISADVANTAGES

- Reward model quality is the bottleneck
- Risk of over-optimizing the reward at the cost of naturalness
- Human evaluation is expensive and slow for iteration

BEST FOR

IMPROVING NATURALNESS AND EXPRESSIVENESS OF TTS

REDUCING AUDIBLE ARTIFACTS IN CODEC-BASED SYNTHESIS

ADAPTING SPEECH MODELS TO SPECIFIC ACOUSTIC ENVIRONMENTS

KEY PAPERS

[1] NaturalSpeech 3: Zero-Shot Speech Synthesis with Factorized Flow — Microsoft, 2024

[2] Voicebox: Text-Guided Multilingual Speech Generation — Meta AI, 2023

VARIANTS

- Preference-based speech RL (human preference pairs for speech)
- Adversarial RL for speech (discriminator as reward)

OPEN SOURCE

`Voicebox (https://github.com/facebookresearch/voicebox)`

`NaturalSpeech (https://github.com/microsoft/NaturalSpeech)`

COMMON MISTAKES

- *Using a MOS prediction model that does not correlate with human judgment*
- *Applying RL too early (before SFT convergence)*
- *Not preserving speaker identity during RL optimization*

FUTURE DIRECTIONS

Multimodal speech RL that jointly optimizes for intelligibility, naturalness, emotion expressiveness, and speaker similarity using a multi-task reward model.

RECIPE 04

Multi-Speaker Distillation

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 4–16 GPUs for 3–10 days depending on dataset size and model architecture

PURPOSE

Train a single speech model to handle multiple speakers by distilling speaker-specific characteristics into a unified model, enabling multi-speaker TTS without per-speaker fine-tuning.

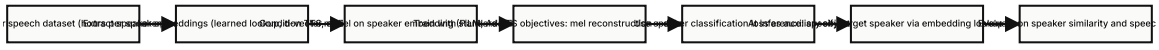
USED BY

YOURTTS | TACOTRON MULTI-SPEAKER | VITS MULTI-SPEAKER | XLISP

CORE IDEA

Multi-speaker distillation trains a single model to generate speech for many different speakers by conditioning on a speaker embedding. The speaker embedding can be a learned lookup table (for fixed speaker sets) or a speaker encoder network trained to extract speaker characteristics from a short reference audio (for unseen speakers). Training data consists of speech from many speakers with speaker labels or reference audio. The model learns to disentangle content (what is said) from speaker identity (who is saying it).

TRAINING PIPELINE



ADVANTAGES

- Single model for many speakers — efficient deployment
- Zero-shot cloning from short reference audio
- Can interpolate between speakers for novel voices

DISADVANTAGES

- Speaker identity leakage into content representation
- Quality degrades for speakers far from the training distribution
- Requires careful data curation for balanced speaker representation

BEST FOR

MULTI-SPEAKER TTS PRODUCTS AND SERVICES

ZERO-SHOT VOICE CLONING FROM SHORT SAMPLES

AUDIOBOOK AND NARRATION GENERATION

KEY PAPERS

[1] YourTTS: Towards Zero-Shot Multi-Speaker TTS — Casanova et al., 2021

[2] VITS: Conditional Variational Autoencoder with Adversarial Learning — Kim et al., 2021

[3] Speaker Embedding Extraction for Multi-Speaker TTS — Various, 2023

VARIANTS

- Speaker-adaptive training (fine-tune a base model to new speakers)
- Speaker-conditional flow matching (natural flow TTS)

OPEN SOURCE

`Coqui TTS (https://github.com/coqui-ai/TTS)`
`VITS (https://github.com/jaywalnut310/vits)`
`YourTTS (https://github.com/Edresson/YourTTS)`

COMMON MISTAKES

- *Having too few speakers (model overfits to specific voices)*
- *Not balancing speaker representation in training data*
- *Using speaker embeddings that are too high-dimensional (overfitting)*

FUTURE DIRECTIONS

Universal speaker model that can clone any voice from a 1-second sample and generate speech in any language with consistent speaker identity, using a large-scale multi-speaker multi-lingual pre-training approach.

RECIPE 05

Voice Cloning

COMPLEXITY

★★★☆☆

COMPUTE

Low-Medium — 1–8 GPUs for 1–7 days for training the base model; seconds for zero-shot inference

PURPOSE

Replicate a target speaker's voice characteristics from a limited reference sample (few-shot or zero-shot), enabling personalized speech synthesis.

USED BY

VALL-E | BARK (SUNO) | XTTS (COQUI) | OPENVOICE

CORE IDEA

Voice cloning adapts a generic TTS model to a target speaker using minimal reference audio. Zero-shot methods (VALL-E, XTTS) use a speaker encoder that extracts a voice embedding from the reference and conditions the TTS model at inference without any fine-tuning. Few-shot methods use a short adaptation step (1-30 seconds of audio) to fine-tune a base model. The key challenges are maintaining naturalness while accurately reproducing the target voice, and preventing speaker leakage from the reference into the content.

TRAINING PIPELINE

ADVANTAGES

- Personalized speech from very limited reference data
- Zero-shot methods require no per-speaker training
- Improving rapidly with scale and better architectures

DISADVANTAGES

- Speaker similarity degrades with very short (< 3s) reference audio
- Risk of misuse (voice spoofing, deepfake audio)
- Less natural than speaker-specific fine-tuned models

BEST FOR

PERSONALIZED VOICE ASSISTANTS

AUDIOBOOK NARRATION IN A SPECIFIC VOICE

CONTENT CREATION AND DUBBING

VARIANTS

- Cross-lingual voice cloning (clone in one language, generate in another)
- Emotion-preserving voice cloning (clone voice + emotion from reference)

KEY PAPERS

[1] VALL-E: Neural Codec Language Models for Text-to-Speech — Wang et al. (Microsoft), 2023

[2] OpenVoice: Versatile Instant Voice Cloning — Qin et al., 2023

[3] XTTS: Cross-Lingual Zero-Shot TTS — Coqui AI, 2024

OPEN SOURCE

`Coqui TTS (https://github.com/coqui-ai/TTS)`

`XTTS (https://github.com/coqui-ai/TTS)`

`OpenVoice (https://github.com/myshell-ai/OpenVoice)`

COMMON MISTAKES

- *Using too little reference audio (< 3 seconds for good quality)*
- *Generating in a language/accent the base model was not trained on*
- *Not applying anti-spoofing safeguards in deployment*

FUTURE DIRECTIONS

Real-time voice cloning with emotional and prosodic control, where the cloned voice can laugh, whisper, shout, and convey nuanced emotions — indistinguishable from a real human recording.

PART 5

Robotics

7 recipes from latent world models (Dreamer) and action diffusion through sim-to-real transfer, behavior cloning, offline RL, and interactive correction loops — covering the full training stack for embodied AI.

RECIPE 01 | PART ROBOTICS

World Models

COMPLEXITY

★★★★☆

COMPUTE

Medium — 1–8 GPUs for 2–7 days for typical benchmarks; scales with environment complexity

PURPOSE

Train a latent dynamics model of the environment that enables a policy to learn by "imagining" future trajectories through the learned world model, reducing real-world interaction.

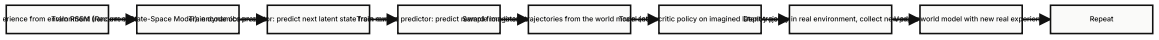
USED BY

DREAMERV3 (DEEPMIND) | DAYDREAMER | IRIS | TRANSDREAMER

CORE IDEA

World models learn a compressed latent representation of the environment dynamics, then train a policy entirely within this latent space. DreamerV3 encodes observations into a compact latent state, predicts future latent states and rewards using a recurrent dynamics model, and trains an actor-critic policy on latent imaginary rollouts. Because the model "dreams" trajectories, the policy can learn from far more experience than was actually collected in the real environment. This dramatically improves sample efficiency.

TRAINING PIPELINE



ADVANTAGES

- Extremely sample efficient — learns from 1% of the data model-free methods need
- Learns a compact representation of environment dynamics
- Policy training is fast (no environment latency)
- Works with sparse rewards due to dense imagined training

DISADVANTAGES

- World model errors compound during long-horizon imagination
- Requires careful tuning of the latent dynamics model
- Limited to environments that can be accurately modeled

BEST FOR

CONTINUOUS CONTROL TASKS

ENVIRONMENTS WITH EXPENSIVE OR SLOW REAL-WORLD INTERACTION

TASKS REQUIRING LONG-HORIZON PLANNING

REAL-WORLD ROBOTICS (SAMPLE EFFICIENCY MATTERS)

VARIANTS

- TransDreamer (transformer-based world model for better long-horizon prediction)
- IRIS (discrete autoencoder + transformer world model for Atari)
- DayDreamer (on-robot world model training for real-world robotics)

KEY PAPERS

[1] Dream to Control: Learning Behaviors by Latent Imagination — Hafner et al. (DeepMind), 2019

[2] Mastering Diverse Domains through World Models (DreamerV3) — Hafner et al. (DeepMind), 2023

[3] DayDreamer: World Models for Physical Robot Learning — Wu et al., 2022

OPEN SOURCE

`DreamerV3 (https://github.com/danijar/dreamerv3)`

`Dreamer (https://github.com/google-research/dreamer)`

COMMON MISTAKES

- *Using a world model that is too small to capture environment dynamics*
- *Not training the world model long enough before policy training*
- *Imagining trajectories that are too short (policy does not learn planning)*

FUTURE DIRECTIONS

Large-scale pre-trained world models that capture general physics and dynamics from internet video, enabling zero-shot planning in novel environments without any environment interaction.

RECIPE 02

Action Diffusion

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 1–8 GPUs for 2–7 days for typical real-world policy training

PURPOSE

Generate robot action sequences using diffusion models, enabling smooth, multimodal, and temporally consistent behavior generation for complex manipulation and locomotion tasks.

USED BY

DIFFUSION POLICY | CHAINED DIFFUSION | ROBOTICS DIFFUSION MODELS

CORE IDEA

Action diffusion treats action generation as a denoising process: starting from random noise in action space, a diffusion model iteratively denoises toward a valid action sequence. The model is conditioned on visual observations and (optionally) goal states. The key advantages over direct regression are multimodality (multiple valid actions for the same observation) and temporal consistency (actions are generated as a sequence, not per-timestep). This produces smoother, more robust robot behavior.

TRAINING PIPELINE

ADVANTAGES

- Inherently multimodal — produces diverse valid actions
- Smooth action sequences (avoids jerky per-timestep noise)
- Robust to observation noise through the denoising process
- Easy to incorporate via pre-trained diffusion backbones

DISADVANTAGES

- Slower than direct regression (multiple denoising steps)
- Requires large demonstration datasets
- Diffusion inference is more complex than standard policy forward passes

BEST FOR

VISUOMOTOR POLICY LEARNING (IMAGE → ACTIONS)

FINE MANIPULATION (SMOOTH, PRECISE MOVEMENTS NEEDED)

TASKS WITH DIVERSE ACTION MODALITIES

KEY PAPERS

[1] Diffusion Policy: Visuomotor Policy Learning via Action Diffusion — Chi et al., 2023

[2] Chained Diffusion Models for Robotic Manipulation — Various, 2023

[3] Generalized Diffusion Policy for Diverse Robot Tasks — Zhou et al., 2024

VARIANTS

- Observation-space diffusion (diffuse over observation latents + actions jointly)
- Hierarchical action diffusion (high-level plan → low-level actions)

OPEN SOURCE

`Diffusion Policy (https://github.com/real-stanford/diffusion_policy)`

COMMON MISTAKES

- *Using too short a temporal window (loses consistency benefits)*
- *Too many denoising steps at inference (latency)*
- *Not tuning the observation conditioning properly*

FUTURE DIRECTIONS

Real-time action diffusion on robot hardware with 1-2 denoising steps (consistency-style), achieving diffusion policy quality at control-loop frequency.

RECIPE 03

Latent Planning

COMPLEXITY

★★★★☆

COMPUTE

Medium — 1–4 GPUs for 2–7 days for visual control benchmarks

PURPOSE

Plan action sequences in a learned latent space rather than raw observation or action space, enabling efficient long-horizon planning by compressing high-dimensional information.

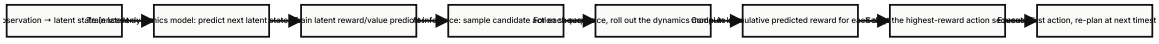
USED BY

TD-MPC2 | PLAN2EXPLORE | TAP (TASK AND MOTION PLANNING IN LATENT)

CORE IDEA

Latent planning learns a compressed latent representation of the world state, then plans action sequences within this latent space. The planner searches over action sequences by simulating their outcomes in the learned latent dynamics model and selecting the sequence that maximizes predicted reward. TD-MPC2 uses a latent representation jointly trained for reconstruction, reward prediction, and task-relevant features. Planning in latent space is faster and more sample-efficient than planning in pixel space because the latent representation strips away irrelevant visual details.

TRAINING PIPELINE



ADVANTAGES

- Sample efficient — learns from 0.1-1M environment steps
- Long-horizon planning capability
- Faster than planning in pixel space
- Handles high-dimensional observations (images, point clouds)

DISADVANTAGES

- Planning horizon limited by dynamics model accuracy
- Latent representation may discard task-relevant details
- MPC planning adds inference-time compute

BEST FOR

VISUAL CONTROL TASKS WITH HIGH-DIMENSIONAL OBSERVATIONS

LONG-HORIZON MANIPULATION TASKS

TASKS WHERE REWARD IS SPARSE (PLANNING COMPENSATES)

VARIANTS

- Hierarchical latent planning (plan in abstract latent, refine in detail)
- Goal-conditioned latent planning (plan paths to goal states)

KEY PAPERS

[1] TD-MPC2: Accelerated Model-Based Reinforcement Learning — Hansen et al., 2023

[2] Planning to Explore via Self-Supervised World Models — Sekar et al., 2020

OPEN SOURCE

`TD-MPC2 (https://github.com/nicklashansen/td-mpc2)`

`Dreamer (https://github.com/google-research/dreamer)`

COMMON MISTAKES

- *Planning horizon too short (myopic behavior)*
- *Not using a sufficiently expressive latent representation*
- *Too few planning candidates — misses good sequences*

FUTURE DIRECTIONS

Pre-trained latent planning modules from internet video that provide a "universal latent dynamics" for any task, enabling zero-shot planning in novel environments.

RECIPE 04

Sim-to-Real Transfer

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 1–8 GPUs for 1–7 days in simulation; real-world deployment cost varies

PURPOSE

Train robot policies in simulation and transfer them to real hardware, leveraging simulation data abundance while overcoming the domain gap between simulated and real environments.

USED BY

DOMAIN RANDOMIZATION | SIM-TO-REAL RL (OPENAI DACTYL) | ISAAC GYM TRAINING

CORE IDEA

Sim-to-real transfer trains large quantities of experience in simulation (where data is cheap and fast) and then adapts the policy for real-world deployment. The key technique is domain randomization: systematically randomizing simulation parameters (mass, friction, lighting, textures) so the policy learns to generalize across the sim-to-real gap rather than overfitting to specific simulation values. More advanced methods use system identification (matching sim parameters to real observations) or domain adaptation (learning invariant features).

TRAINING PIPELINE

ADVANTAGES

- Millions of training steps in hours instead of months
- Safe exploration — no hardware damage risk
- Covers edge cases that are rare or dangerous in reality

DISADVANTAGES

- Domain gap limits final performance
- Randomization must cover the right parameters
- Simulation fidelity is expensive to build

BEST FOR

ANY ROBOT LEARNING TASK WHERE SIMULATION IS AVAILABLE

MANIPULATION AND LOCOMOTION TASKS

DEPLOYING RL POLICIES ON REAL HARDWARE

VARIANTS

- System identification (calibrate simulation to match real observations)
- Domain adaptation (learn features invariant to the sim/real domain)
- Progressive net (learn residual on real data after sim pre-training)

KEY PAPERS

[1] Learning Dexterous In-Hand Manipulation (OpenAI Dactyl) — OpenAI, 2018

[2] Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World — Tobin et al. (OpenAI), 2017

[3] Sim-to-Real Transfer in Deep Reinforcement Learning — Various (Survey), 2023

OPEN SOURCE

`Isaac Gym (https://github.com/NVIDIA-Omniverse/IsaacGymEnvs)`

`MuJoCo (https://github.com/deepmind/mujoco)`

`SAPIEN (https://github.com/haosulab/SAPIEN)`

COMMON MISTAKES

- *Over-randomizing — policy learns random behavior instead of the task*
- *Not randomizing the right parameters (visual ≠ physical)*
- *Assuming the sim perfectly models the real system*

FUTURE DIRECTIONS

Foundational sim-to-real: a single large-scale simulation training run that produces policies capable of zero-shot deployment on diverse real robot platforms without per-platform fine-tuning.

RECIPE 05

Behavior Cloning

COMPLEXITY

★★☆☆☆

COMPUTE

Low-Medium — 1–8 GPUs for 1–4 days depending on model and dataset size

PURPOSE

Learn robot control policies by directly imitating expert demonstrations, treating the problem as supervised learning: map observations to actions.

USED BY

ALOHA (MOBILE ALOHA) | RT-2 (GOOGLE) | IMITATION LEARNING PIPELINES

CORE IDEA

Behavior cloning frames robot learning as a supervised problem: given a dataset of expert demonstrations (observation → action pairs), train a neural network to predict the expert action from the observation. While conceptually simple, practical BC requires addressing distribution shift (the policy encounters states not in the training data), action stochasticity, and demonstration quality. Modern BC for robotics uses large vision-language backbones (RT-2), diffusion-based action prediction, and data augmentation to handle multimodal action distributions.

TRAINING PIPELINE

ADVANTAGES

- Simple supervised learning — no reward design
- Fast to train compared to RL methods
- Safe (imitates expert behavior, avoids random exploration)

DISADVANTAGES

- Distribution shift at test time (states not seen in training)
- Quality limited by demonstration quality
- Cannot exceed expert performance (unlike RL)

BEST FOR

SMALL-SCALE, SHORT-HORIZON MANIPULATION TASKS

TASKS WITH GOOD DEMONSTRATION COVERAGE

STARTING POINT BEFORE RL FINE-TUNING

KEY PAPERS

[1] Mobile ALOHA: Learning Bimanual Mobile Manipulation — Fu et al., 2023

[2] RT-2: Vision-Language-Action Models for Web-Scale Robot Control — Zitkovich et al. (Google DeepMind), 2023

[3] A Survey of Imitation Learning for Robotics — Various, 2022

VARIANTS

- DAgger (Dataset Aggregation — collect new demos at states visited by current policy)
- Visual behavior cloning (image → actions with vision backbone)
- Diffusion behavior cloning (diffusion over action sequences)

OPEN SOURCE

`Mobile ALOHA (https://github.com/MarkFzp/mobile-aloha)`
`RLbench (https://github.com/stepjam/RLBench)`

COMMON MISTAKES

- *Training on too few demonstrations (does not cover state space)*
- *Not using any noise injection or augmentation (brittle policy)*
- *Using deterministic action prediction for inherently multimodal tasks*

FUTURE DIRECTIONS

Internet-scale behavior cloning where robot policies are pre-trained on human video datasets showing diverse manipulation, enabling zero-shot generalization to new tasks and objects.

RECIPE 06

Offline RL

COMPLEXITY

★★★★☆

COMPUTE

Medium — 1–4 GPUs for 1–5 days depending on dataset size

PURPOSE

Learn optimal policies entirely from previously collected, static datasets without any environment interaction — enabling robot learning from pre-existing log data.

USED BY

CONSERVATIVE Q-LEARNING (CQL) | IQL | EDAC | TD3+BC

CORE IDEA

Offline RL trains policies from static datasets collected by any behavioral policy (human demonstrations, deployed robots, random exploration). The key challenge is distribution shift: the learned policy will encounter state-action pairs not present in the dataset, and standard RL overestimates Q-values for unseen actions. Solutions include conservative Q-learning (penalizing Q-values for out-of-distribution actions), implicit Q-learning (avoiding querying unseen actions entirely), and advantage-weighted regression (filtering by action quality).

TRAINING PIPELINE

ADVANTAGES

- Uses existing data — no environment interaction needed during training
- Safe — no exploration of dangerous states during learning
- Can leverage terabytes of previously collected robot data

DISADVANTAGES

- Distribution shift is a fundamental challenge
- Quality ceiling is limited by the dataset
- Evaluation requires a simulator or real deployment

BEST FOR

ROBOT LEARNING FROM LOGGED INTERACTION DATA

DOMAINS WHERE ENVIRONMENT INTERACTION IS EXPENSIVE OR RISKY

PRE-TRAINING A POLICY FROM DIVERSE DATA BEFORE ONLINE FINE-TUNING

VARIANTS

- Offline-to-online (pre-train offline, fine-tune with minimal online interaction)
- Decision Transformer (trajectory-level supervised offline RL)

KEY PAPERS

[1] Conservative Q-Learning for Offline Reinforcement Learning — Kumar et al., 2020

[2] Implicit Q-Learning: Offline Reinforcement Learning via the Advantage — Kostrikov et al., 2021

[3] A Minimalist Approach to Offline Reinforcement Learning (TD3+BC) — Fujimoto & Gu, 2021

OPEN SOURCE

`d3rlpy (https://github.com/takuseno/d3rlpy)`

`CORL (https://github.com/tinkoff-ai/CORL)`

COMMON MISTAKES

- *Using offline RL on data collected by a very different policy than the target*
- *Not tuning the conservatism penalty (CQL alpha is critical)*
- *Forgetting that evaluation still requires a simulator or real system*

FUTURE DIRECTIONS

Large-scale offline RL datasets for robotics (like D4RL scaled 1000x) combined with foundation model backbones, enabling generalist robot policies that can be deployed zero-shot then fine-tuned with minimal online interaction.

RECIPE 07

Interactive Learning

COMPLEXITY

★★★☆☆

COMPUTE

Low — 1 GPU for policy updates; human interaction is the bottleneck

PURPOSE

Continuously improve robot policies through online human correction signals — where a human supervisor provides corrective feedback that the policy uses to improve without full re-training.

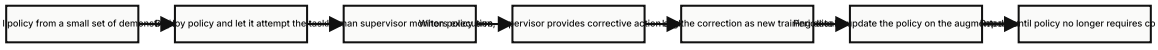
USED BY

DAGGER | HG-DAGGER | INTERACTIVE IMITATION LEARNING | ROBOT CORRECTIVE FEEDBACK

CORE IDEA

Interactive learning combines the safety of imitation learning with the improvement capability of RL. A human supervisor watches the policy execute and provides corrective interventions or demonstrations when the policy makes mistakes (DAgger). These corrections are aggregated into the training dataset, and the policy is updated to avoid the corrected mistakes. This cycle continues until the policy achieves desired performance. Modern variants use only corrective feedback (not full demonstrations) and can handle high-frequency intervention.

TRAINING PIPELINE



ADVANTAGES

- Policies improve beyond initial demonstration quality
- Safer than pure RL (human monitors and corrects)
- Requires less human effort than full demonstrations

DISADVANTAGES

- Requires human in the loop during training
- Human correction latency limits task complexity
- Correction variance — different humans correct differently

BEST FOR

TASKS WHERE DEMONSTRATIONS ARE AVAILABLE BUT IMPERFECT

DEPLOYING SAFE INITIAL POLICIES THAT IMPROVE OVER TIME

LONG-HORIZON TASKS WHERE FULL DEMOS ARE IMPRACTICAL

KEY PAPERS

[1] A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning — Ross et al. (DAgger), 2011

[2] Human-Guided DAgger for Robot Manipulation — Various, 2023

[3] Interactive Imitation Learning from Visual Corrections — Various, 2024

VARIANTS

- HG-DAgger (human-gated correction collection)
- Corrective feedback only (no full demos, just adjustments)
- Confidence-based querying (policy asks for help when uncertain)

OPEN SOURCE

`Various (task-specific implementations)`

COMMON MISTAKES

- *Collecting too few corrections before updating (overfits to recent corrections)*
- *Not normalizing corrective action distributions*
- *Human supervisor correcting too aggressively (confusing the policy)*

FUTURE DIRECTIONS

Scalable interactive learning where corrective feedback is provided by foundation models (LLMs that detect policy errors) instead of humans, enabling 24/7 automated policy improvement.

PART 6

Agents

7 recipes building from tool-use RL through web agents and computer-use models, memory optimization, skill distillation, hierarchical planning, and multi-agent coordination — the practical training stack for deploying LLM agents.

RECIPE 01 PART AGENTS

Tool-Use RL

COMPLEXITY

★★★★☆

COMPUTE

Medium-High — 8–32 GPUs for 5–14 days including tool execution infrastructure

PURPOSE

Train language models to autonomously decide when and how to use external tools (APIs, calculators, search, code interpreters) through reinforcement learning from tool interaction outcomes.

USED BY

GPT-4 WITH TOOLS | CLAUDE TOOL USE | GEMINI FUNCTION CALLING | TOOLFORMER

CORE IDEA

Tool-use RL trains models to generate tool calls (function invocations) in addition to text. The model outputs structured tool calls, executes them, receives the result, and continues generating with the tool output in context. Training uses outcome-based rewards (did the tool use help solve the task?), which naturally handles the exploration-exploitation tradeoff of tool selection. The training loop interleaves text generation with tool execution, and the policy gradient rewards successful tool-use trajectories.

TRAINING PIPELINE

ADVANTAGES

- Model learns when to use each tool, not just how
- Can discover novel tool-use strategies not in demonstrations
- Generalizes to new tools with similar schemas

DISADVANTAGES

- Sandboxed tool execution is complex infrastructure
- Long latency — each tool call blocks generation
- Hard to credit-assign across multiple tool calls

BEST FOR

MATH AND CALCULATION (CALCULATOR TOOL)

INFORMATION RETRIEVAL (SEARCH, DATABASE QUERIES)

CODE GENERATION AND EXECUTION (INTERPRETER TOOL)

API ORCHESTRATION AND AUTOMATION

VARIANTS

- Tool-augmented language models (always call tools, no RL — simpler)
- Self-instruct for tools (model generates its own tool-use demonstrations)

KEY PAPERS

[1] Toolformer: Language Models Can Teach Themselves to Use Tools — Schick et al. (Meta AI), 2023

[2] Gorilla: Large Language Model Connected with Massive APIs — Patil et al., 2023

[3] Tool Use in Large Language Models: A Survey — Various, 2024

OPEN SOURCE

`Toolformer (https://github.com/lucidrains/toolformer-pytorch)`

`Gorilla (https://github.com/ShishirPatil/gorilla)`

`OpenAI Function Calling examples`

COMMON MISTAKES

- *Sandbox not properly isolated (tool execution security)*
- *Reward shaping that over-penalizes tool calls (model stops using tools)*
- *Not handling tool execution errors gracefully during training*

FUTURE DIRECTIONS

Meta-tool-use: models that can create and register new tools on the fly by writing code, testing it, and incorporating it into their tool set during a single session.

RECIPE 02

Web Agents

COMPLEXITY

★★★★★

COMPUTE

High — 16–64 GPUs for 7–21 days; significant infrastructure for browser environment

PURPOSE

Train language models to navigate and interact with web interfaces (browsers, forms, search) autonomously, treating web navigation as a partially-observed sequential decision problem.

USED BY

WEBGPT (OPENAI) | WEBVOYAGER | MIND2WEB | AGENTQL

CORE IDEA

Web agents treat browser interaction as a sequential decision task. The model observes the web page state (HTML, accessibility tree, or screenshots), selects actions (click, type, scroll, navigate), observes the resulting page state, and continues until the task is complete. Training uses a combination of behavior cloning from human web demonstrations and RL from task completion rewards. Modern web agents use vision-language models to process screenshots directly, eliminating the need for HTML parsing.

TRAINING PIPELINE

ADVANTAGES

- Can automate any web-based workflow
- VLM-based agents work on any visual interface
- Improves with scale (more data, larger models)

DISADVANTAGES

- Slow execution (each action requires model inference)
- Brittle to website layout changes
- Safety concerns (autonomous web navigation)

BEST FOR

FORM FILLING AND DATA ENTRY AUTOMATION

WEB RESEARCH AND INFORMATION GATHERING

E-COMMERCE AND BOOKING AUTOMATION

VARIANTS

- VLM-based web agents (screenshot → actions via vision)
- HTML-accessible tree agents (text-only for efficiency)

KEY PAPERS

[1] WebGPT: Browser-Assisted Question-Answering — OpenAI, 2021

[2] Mind2Web: Towards a Generalist Web Agent — Deng et al., 2023

[3] WebVoyager: Building an End-to-End Web Agent — He et al., 2024

OPEN SOURCE

`WebVoyager (https://github.com/webvoyager/webvoyager)`
`Playwright (https://github.com/microsoft/playwright)`
`BrowserGym (https://github.com/ServiceNow/BrowserGym)`

COMMON MISTAKES

- *Not handling dynamic content (loading spinners, AJAX updates)*
- *Using HTML-only representation (misses visual layout information)*
- *Task descriptions too vague for reliable success detection*

FUTURE DIRECTIONS

Self-improving web agents that automatically recover from failures, learn from their mistakes across sessions, and build internal models of common website patterns.

RECIPE 03

Computer-Use Models

COMPLEXITY

★★★★★

COMPUTE

Very High — 32–128 GPUs for 14–30 days; heavy infrastructure for interaction environment

PURPOSE

Train models to interact with computer interfaces at the pixel level — moving the cursor, clicking, typing — treating any GUI application as a controllable environment.

USED BY

CLAUDE COMPUTER USE (ANTHROPIC) | UI-SELECTOR | SCREENAGENT | PIXELACTOR

CORE IDEA

Computer-use models operate directly on screen pixels: they take a screenshot as input, decide where to move the cursor and what action to take (click, right-click, type, scroll), and receive the updated screenshot as the next observation. This is a pixel-level agent that works on any GUI without API access. Training combines behavior cloning from human computer-use traces (mouse movements, clicks) with RL from task completion. The pixel-level approach generalizes across operating systems and applications.

TRAINING PIPELINE

ADVANTAGES

- Works on any GUI — no API needed
- Generalizes across applications and operating systems
- Can automate legacy software without automation APIs

DISADVANTAGES

- Slow — each action requires a full model forward pass
- Pixel coordinate precision is challenging
- High compute cost per task completion

BEST FOR

LEGACY SOFTWARE AUTOMATION | CROSS-PLATFORM GUI AUTOMATION | SOFTWARE TESTING AND QA | PERSONAL ASSISTANT AUTOMATION

KEY PAPERS

[1] Claude Computer Use (Anthropic) — Anthropic, 2024

[2] ScreenAgent: A Vision Language Model for Computer Control — Various, 2024

[3] PixelActor: Training Pixel-Based Computer Use Agents — Various, 2024

VARIANTS

- Accessibility-tree agents (use OS accessibility API instead of pixels)
- Hybrid pixel + HTML agents (use best available representation)

OPEN SOURCE

`Claude Computer Use (https://github.com/anthropics/claude-computer-use)`

`CogAgent (https://github.com/THUDM/CogAgent)`

COMMON MISTAKES

- *Using too low resolution screenshots (model misses small UI elements)*
- *Not handling screen coordinate system changes (scaling, multi-monitor)*
- *No confirmation step for destructive actions (delete, send)*

FUTURE DIRECTIONS

Computer-use agents with long-term memory of application layouts and workflows, enabling them to build and reuse "app skills" across sessions — learning Photoshop once, applying everywhere.

RECIPE 04

Memory Optimization

COMPLEXITY

★★★★☆

COMPUTE

Medium — 4–16 GPUs for 3–10 days depending on memory architecture

PURPOSE

Train agents to efficiently store, retrieve, and update information across long interactions using learned memory mechanisms — including RAG fine-tuning, memory consolidation, and context compression.

USED BY

- MEMGPT (MEMWALKER)
- RAG FINE-TUNED LMS
- MEMORY-AUGMENTED AGENTS

CORE IDEA

Memory optimization trains agents to manage their own context window: deciding what to remember, what to forget, and how to retrieve relevant information when needed. Key techniques include fine-tuning for retrieval-augmented generation (RAG) where the model learns to condition on retrieved documents effectively; memory consolidation where short-term memories are summarized into long-term storage; and context compression where long histories are distilled into compact representations.

TRAINING PIPELINE

ADVANTAGES

- Enables agents to maintain coherence over extremely long interactions
- Reduces context window pressure
- Learns what information is worth remembering

DISADVANTAGES

- Memory systems add latency to each agent step
- Consolidation can lose important details
- Retrieval quality is a bottleneck

BEST FOR

- LONG-RUNNING AGENT SESSIONS (HOURS OR DAYS)
- PERSONAL ASSISTANTS THAT BUILD USER MODELS OVER TIME
- RESEARCH AGENTS CONDUCTING MULTI-DAY INVESTIGATIONS

VARIANTS

- Hierarchical memory (short-term → episodic → semantic consolidation)
- Compressive memory (train a smaller model to compress experiences)

KEY PAPERS

[1] MemGPT: Towards LLMs as Operating Systems — Packer et al., 2023

[2] Retrieval-Augmented Generation for Knowledge-Intensive Tasks — Lewis et al., 2020

[3] Unlimiformer: Long-Range Transformers with Unlimited Length — Bertsch et al., 2023

OPEN SOURCE

`MemGPT (https://github.com/cpacker/MemGPT)`

`LangChain memory integrations`

COMMON MISTAKES

- *Storing too much — retrieval quality degrades with dense memory stores*
- *Not training the retrieval module jointly with the agent*
- *Consolidating too aggressively — losing important episodic details*

FUTURE DIRECTIONS

Agents with infinite context windows achieved through learned compression, where the model itself decides at each token whether to store, retrieve, or discard information — transparent and inspectable.

RECIPE 05

Skill Distillation

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 4–16 GPUs for 2–7 days depending on teacher size and data volume

PURPOSE

Compress agentic workflows and decision-making capabilities from large, expensive models (or human demonstrations) into smaller, faster models for efficient deployment.

USED BY

SMALL AGENT MODELS (PHI-3 AGENTS) | TOOL-USE DISTILLATION | AGENT COMPRESSION

CORE IDEA

Skill distillation transfers agentic capabilities from a capable but expensive teacher (large LM or human) to a smaller student model. The student learns to imitate the teacher action sequences, tool-use decisions, and planning trajectories. The training data is generated by the teacher executing tasks and logging its full decision process. In addition to action imitation, the student learns the teacher intent — why it chose specific actions — through intermediate reasoning distillation. This enables deployable agents at a fraction of the cost.

TRAINING PIPELINE

ADVANTAGES

- Dramatically reduces inference cost (10-100x cheaper)
- Lower latency — critical for real-time agent interactions
- Can distill specific skills (tool use, web nav, planning) independently

DISADVANTAGES

- Student quality capped by teacher quality
- Risk of overfitting to teacher failure modes
- Requires careful trajectory curation and filtering

BEST FOR

DEPLOYING AGENTIC CAPABILITIES AT SCALE

EDGE AND MOBILE AGENT DEPLOYMENT

SPECIALIZED AGENT SKILLS (CALCULATOR, SEARCH, CODE)

VARIANTS

- Multi-teacher distillation (distill from multiple teacher models)
- Progressive distillation (start with small teacher, increase over rounds)

KEY PAPERS

[1] Distilling Agentic Capabilities into Smaller Models — Various, 2024

[2] Tool Distillation: Compressing Tool-Use into Small LMs — Various, 2024

[3] A Survey of Model Distillation for LLMs — Various, 2024

OPEN SOURCE

`Axolotl (https://github.com/axolotl-ai-cloud/axolotl)`

`LLM Distributed (https://github.com/EleutherAI/llm-distributed)`

COMMON MISTAKES

- *Distilling from a teacher that is not sufficiently capable in the target skill*
- *Not filtering low-quality teacher trajectories*
- *Using too small a student model (cannot absorb the distilled skill)*

FUTURE DIRECTIONS

Lifelong distillation where agents continuously distill new skills from experience into a growing library of specialized small models, each handling one capability with expert-level performance.

RECIPE 06

Hierarchical Planning

COMPLEXITY

★★★★☆

COMPUTE

Medium — 4–16 GPUs for 5–14 days depending on hierarchy depth

PURPOSE

Train agents to decompose complex tasks into hierarchical subgoals, plan at multiple levels of abstraction, and execute plans through lower-level policies — enabling long-horizon task completion.

USED BY

HUGGINGGPT (TRANSFORMERS AGENT) | VOYAGER (MINEDOJO) | PLAN-AND-EXECUTE AGENTS

CORE IDEA

Hierarchical planning trains agents to operate at multiple levels of abstraction. A high-level planner decomposes a task into subgoals (a plan), and lower-level policies execute each subgoal. The high-level planner receives the overall task and the current world state, and outputs a sequence of subgoals. Each subgoal is passed to a lower-level policy trained to achieve that specific subgoal. Training alternates between high-level plan optimization and low-level skill refinement, often using a combination of supervised learning on demonstrations and RL on task completion.

TRAINING PIPELINE

ADVANTAGES

- Handles long-horizon tasks that flat policies cannot
- More interpretable — plans can be inspected at each level
- Skills are reusable across different tasks (composability)

DISADVANTAGES

- Planning horizon is limited by the highest-level planner
- Error propagation: a bad plan fails even with good skills
- Training is more complex than flat policy training

BEST FOR

LONG-HORIZON SOFTWARE ENGINEERING TASKS

MULTI-STEP WEB RESEARCH AND DATA COLLECTION

ROBOTICS TASK AND MOTION PLANNING

VARIANTS

- Dynamic hierarchical planning (hierarchy depth adapts to task complexity)
- Recursive planning (task → subtask → sub-subtask recursively)

KEY PAPERS

[1] HuggingGPT: Solving AI Tasks with ChatGPT and its Friends — Shen et al., 2023

[2] Voyager: An Open-Ended Embodied Agent with Large Language Models — Wang et al. (NVIDIA), 2023

[3] Plan-and-Solve: Improving LLM Planning with Explicit Subgoals — Wang et al., 2023

OPEN SOURCE

`Voyager (https://github.com/MineDojo/Voyager)`

`LangChain Plan-and-Execute`

COMMON MISTAKES

- *Too many hierarchy levels (latency and error propagation)*
- *Planner not aware of low-level policy capabilities (unachievable subgoals)*
- *Not providing enough feedback from low-level execution failures back to planner*

FUTURE DIRECTIONS

Meta-learned hierarchies where the agent discovers its own optimal hierarchy for each task domain, automatically identifying the right level of abstraction without human engineering.

RECIPE 07

Multi-Agent RL

COMPLEXITY

★★★★★

COMPUTE

High — 16–128 GPUs for 14–60 days for large-scale MARL training

PURPOSE

Train multiple agents to coordinate, cooperate, or compete effectively through multi-agent reinforcement learning, enabling complex multi-agent systems for software, robotics, and games.

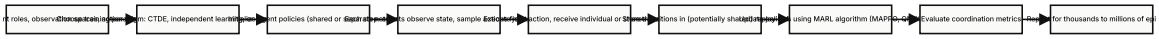
USED BY

OPENAI FIVE | ALPHASTAR | SMAC (STARCRAFT) MARL | AGENT COORDINATION RESEARCH

CORE IDEA

Multi-agent RL extends single-agent RL to settings with multiple simultaneously learning agents. Each agent observes the environment (and optionally other agents), and selects actions. The key challenge is non-stationarity: each agent perceives a changing world because the other agents are also learning. Solutions include centralized training with decentralized execution (CTDE), where agents share gradients during training but act independently at inference; and opponent modeling, where agents learn to predict other agents behavior.

TRAINING PIPELINE



ADVANTAGES

- Can solve tasks requiring coordination (no single agent suffices)
- Emergent behaviors from competitive training
- Parallelized exploration across agents

DISADVANTAGES

- Non-stationarity makes training unstable
- Credit assignment is hard (which agent caused the reward?)
- Exponential blowup in joint action space

BEST FOR

MULTI-AGENT SOFTWARE SYSTEMS (DEV TEAMS OF AGENTS)

GAME AI AND SIMULATION (STARCRAFT, DOTA, HIDE-AND-SEEK)

ROBOTICS SWARM COORDINATION

VARIANTS

- Fully decentralized (each agent learns independently)
- CTDE (centralized training, decentralized execution)
- Shared parameters (all agents use same policy network)
- Mixed cooperative-competitive (some agents cooperate, others compete)

KEY PAPERS

[1] The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games — Yu et al. (MAPPO), 2021

[2] QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent RL — Rashid et al., 2018

[3] Emergent Tool Use from Multi-Agent Interaction (Hide and Seek) — OpenAI, 2019

OPEN SOURCE

`MAPPO (https://github.com/marlbenchmark/on-policy)`

`PyMARL (https://github.com/oxwhirl/pymarl)`

COMMON MISTAKES

- *Using independent learning without CTDE (extreme non-stationarity)*
- *Not sharing rewards in cooperative settings (agents become selfish)*
- *Too many agents for the environment capacity*

FUTURE DIRECTIONS

Foundation model-powered MARL where LLM-based agents negotiate, plan, and coordinate using natural language, enabling complex multi-agent software engineering and scientific research teams.

PART 7

Synthetic Data

7 recipes covering the full synthetic data pipeline: self-instruction, evolutionary expansion, constrained generation, judge/critic training, quality filtering, curriculum synthesis, and production data flywheels for continuous improvement.

RECIPE 01 PART SYNTHETIC-DATA

Self-Instruct

COMPLEXITY

★★☆☆☆

COMPUTE

Low — 1–8 GPUs for 1–3 days; data generation cost equivalent to inference

PURPOSE

Generate large-scale instruction-tuning datasets from a language model itself, bootstrapping instruction-following capability without human-written examples.

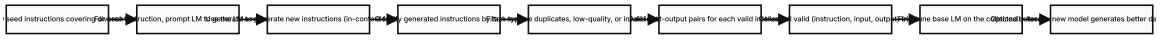
USED BY

SELF-INSTRUCT (SEED INSTRUCTION TUNING) ALPACA (STANFORD) INSTRUCTION-TUNED OPEN MODELS

CORE IDEA

Self-Instruct starts with a small seed set of human-written instructions and uses a language model to generate new instructions, input-output pairs, and diverse task types. The pipeline bootstraps iteratively: the model generates new instructions, filters and validates them, and retrains on the augmented dataset. The generated data covers diverse tasks (classification, generation, reasoning, rewriting) by prompting the model to create examples of each task type. This was the recipe behind Alpaca, Vicuna, and many early open instruction-tuned models.

TRAINING PIPELINE



ADVANTAGES

- Generates unlimited instruction data at minimal cost
- Covers diverse tasks and domains
- Bootstraps instruction following without human annotation

DISADVANTAGES

- Quality ceiling limited by the generating model
- Dataset diversity can plateau without prompt engineering
- Risk of generating factually incorrect or toxic content

BEST FOR

JUMP-STARTING INSTRUCTION-TUNING FOR NEW BASE MODELS

CREATING DIVERSE EVALUATION BENCHMARKS

DOMAIN-SPECIFIC INSTRUCTION DATA GENERATION

KEY PAPERS

[1] Self-Instruct: Aligning Language Models with Self-Generated Instructions — Wang et al., 2022

[2] Stanford Alpaca: An Instruction-Following LLaMA Model — Taori et al., 2023

[3] Self-Instruct Quality: Scaling and Dataset Curation — Various, 2024

VARIANTS

- Cross-model self-instruct (generate with strong model, train on weaker one)
- Iterative self-instruct (improving model generates better data)

OPEN SOURCE

`Self-Instruct (https://github.com/yizhongw/self-instruct)`
`Alpaca-LoRA (https://github.com/tloen/alpaca-lora)`

COMMON MISTAKES

- *Seed set too narrow (generated data lacks diversity)*
- *Not filtering generated instructions (quality issues compound)*
- *Using the same model for generation and training (feedback loops)*

FUTURE DIRECTIONS

Self-instruct at scale using frontier models (Claude, GPT-4) as generators, producing millions of high-quality instruction examples that are then distilled into smaller open models — the recipe behind many of today best open instruction-tuned models.

RECIPE 02

Evol-Instruct

COMPLEXITY

★★★☆☆

COMPUTE

Low-Medium — 1–8 GPUs for 2–5 days; data generation is the main cost

PURPOSE

Evolve simple seed instructions into increasingly complex and diverse training examples through automatic rewriting operations — making instruction-tuned models more capable on hard tasks.

USED BY

WIZARDLM | EVOL-INSTRUCT PIPELINE (MICROSOFT) | OPENORCA DATASET

CORE IDEA

Evol-Instruct starts with a set of basic instructions and uses a language model to apply evolution operations that increase complexity: add constraints, deepen reasoning requirements, convolve multiple tasks, or increase input complexity. Each evolved instruction is validated (does the model still produce a reasonable response?) before being added to the training set. The evolved dataset contains examples at multiple difficulty levels, teaching the model to handle both simple and complex instructions.

TRAINING PIPELINE

ADVANTAGES

- Produces training data at diverse difficulty levels
- Automatically creates hard examples that challenge the model
- Addresses the easy data saturation problem in instruction tuning

DISADVANTAGES

- Evolved instructions can become impossible or contradictory
- Evolution prompt engineering is critical to quality
- Validation step is not perfect — some bad examples slip through

BEST FOR

IMPROVING REASONING AND COMPLEX INSTRUCTION FOLLOWING

MULTI-STEP TASK COMPLETION TRAINING

CREATING CHALLENGING EVALUATION SETS

VARIANTS

- Reverse evol-instruct (start complex, simplify)
- Category-aware evol-instruct (different evolution rates per domain)

KEY PAPERS

[1] WizardLM: Empowering Large Language Models to Follow Complex Instructions — Xu et al. (Microsoft), 2023

[2] Evol-Instruct: Automatic Evolution of Instructions — Microsoft, 2024

[3] OpenOrca: A High-Quality Open Instruction Tuning Dataset — Lian et al., 2024

OPEN SOURCE

`WizardLM (https://github.com/nlpxucan/WizardLM)`

`Evol-Instruct (https://github.com/nlpxucan/evol-instruct)`

COMMON MISTAKES

- *Applying too many evolution rounds (instructions become impossible)*
- *Not validating evolved instructions for solvability*
- *Evolution operations that increase length without increasing difficulty*

FUTURE DIRECTIONS

Automated curriculum evolution where the training process dynamically measures which difficulty levels the model is mastering and evolves more examples at the current frontier of model capability.

RECIPE 03

Constitutional Generation

COMPLEXITY

★★☆☆☆

COMPUTE

Low — 1–8 GPUs for 1–3 days; inference cost dominates

PURPOSE

Generate synthetic training data that respects predefined constraints and quality rubrics — ensuring generated examples meet safety, format, and content standards without manual review.

USED BY

CONSTITUTIONAL DATA GENERATION | SYNTHETIC DATA WITH RUBRICS | GUIDED GENERATION PIPELINES

CORE IDEA

Constitutional generation produces synthetic data by following a written constitution or rubric that specifies constraints on the output. The rubric defines acceptable content, format requirements, safety boundaries, and quality criteria. The generating model receives the rubric as a system prompt when creating each example, and a judge model verifies compliance after generation. This produces training data that is safe, on-format, and high-quality without humans in the loop.

TRAINING PIPELINE

ADVANTAGES

- Generates safe, on-specification data at scale
- Reduces manual data review costs
- Rubrics can be updated without regenerating all data

DISADVANTAGES

- Constitution quality directly determines data quality
- Judge model must be at least as capable as the generator
- Overly restrictive constitutions reduce data diversity

BEST FOR

SAFETY-ALIGNED INSTRUCTION TUNING DATA

FORMAT-SPECIFIC DATA GENERATION (JSON, STRUCTURED OUTPUTS)

DOMAIN-CONSTRAINED DATA (LEGAL, MEDICAL, FINANCIAL)

KEY PAPERS

[1] Constitutional AI: Harmlessness from AI Feedback — Anthropic, 2022

[2] Synthetic Data Generation with Rubrics — Various, 2024

OPEN SOURCE

`Prompt engineering frameworks (LangChain, DSPy)`

VARIANTS

- Iterative constitutional generation (regenerate with feedback from judge)
- Multi-judge constitutional generation (ensemble of judge models)

COMMON MISTAKES

- *Constitution too vague — judge cannot reliably assess compliance*
- *Constitution too specific — all generated data looks identical*
- *Judge model that is weaker than the generator (misses violations)*

FUTURE DIRECTIONS

Self-amending constitutions that identify systematic violations and update the constitution dynamically to close gaps, producing a continuously improving data generation pipeline.

RECIPE 04

Judge Models

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 4–16 GPUs for 3–10 days depending on base model size and data volume

PURPOSE

Train evaluator models that can assess the quality, safety, and correctness of LLM outputs — replacing human evaluation at scale for data filtering, reward modeling, and automated benchmarking.

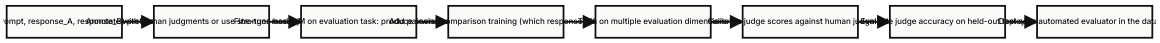
USED BY

GPT-4 AS JUDGE | PAIRRM | ULTRARM | ARMORM | JUDGELM

CORE IDEA

Judge models are LLMs fine-tuned specifically to evaluate the quality of other model outputs. They take a (prompt, response) pair and produce a score, classification, or critique. Training uses human preference data or outputs from stronger models as reference. Modern judge models can assess multiple dimensions (helpfulness, harmlessness, correctness, style) and provide explainable judgments with reasoning. They are used for automated data filtering, as reward models for RLHF, and for benchmarking.

TRAINING PIPELINE



ADVANTAGES

- Replaces expensive human evaluation at scale
- Provides consistent, reproducible evaluations
- Can be updated and improved iteratively

DISADVANTAGES

- Judge bias (prefers outputs similar to its training data)
- Position bias (prefers first or second response)
- Self-enhancement bias (prefers its own architecture)

BEST FOR

AUTOMATED DATA QUALITY FILTERING

REWARD MODEL FOR RLHF/DPO TRAINING

AUTOMATED BENCHMARKING AND LEADERBOARDS

SYNTHETIC DATA VALIDATION PIPELINES

KEY PAPERS

[1] Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena — Zheng et al. (LMSYS), 2023

[2] UltraFeedback: Boosting Language Models with High-Quality Feedback — Cui et al., 2023

[3] ArmoRM: An Adaptive Reward Model for Language Model Alignment — Various, 2024

VARIANTS

- Pairwise judge (which is better, A or B?)
- Absolute judge (rate this output 1-10)
- Critique-based judge (generate explanation + score)

OPEN SOURCE

`UltraFeedback (https://github.com/OpenBMB/UltraFeedback)`

`JudgeLM (https://github.com/baaivision/JudgeLM)`

`PairRM (https://github.com/llm-blender/PairRM)`

COMMON MISTAKES

- *Judge model too weak compared to the models it evaluates*
- *Training on only one dimension (ignoring subtle quality aspects)*
- *Not controlling for position bias in pairwise judgments*

FUTURE DIRECTIONS

Multi-judge ensembles where specialized judges handle different dimensions (factuality, safety, creativity) and a meta-judge arbitrates conflicts — achieving human-level evaluation reliability.

RECIPE 06

Curriculum Generation

COMPLEXITY

★★★☆☆

COMPUTE

Medium — 4–16 GPUs for 3–10 days; generation cost is significant

PURPOSE

Generate synthetic training data at graduated difficulty levels that automatically adjusts to the model current capability, enabling continuous improvement through optimal challenge.

USED BY

PHI-3 CURRICULUM (MICROSOFT) | SELF-PLAY CURRICULUM | AUTOMATIC CURRICULUM RL

CORE IDEA

Curriculum generation creates training examples at multiple difficulty levels and presents them to the model in an order that maximizes learning efficiency. The difficulty of a synthetic example can be controlled by prompt complexity, required reasoning depth, number of constraints, or length of the required response. The curriculum can be static (pre-generated at fixed levels) or dynamic (generated based on the model current performance). Dynamic curriculum uses a zone of proximal development approach: generate examples the model can almost solve but not quite.

TRAINING PIPELINE

ADVANTAGES

- More efficient than uniform-difficulty training
- Prevents overfitting on easy examples early
- Ensures the model is always challenged at the right level

DISADVANTAGES

- Difficulty estimation is not always accurate
- Generating data at the right level requires iteration
- Curriculum pacing is an additional hyperparameter to tune

BEST FOR

PRE-TRAINING SMALL MODELS (PHI-3 APPROACH)

REASONING CAPABILITY IMPROVEMENT

DOMAIN-SPECIFIC CAPABILITY BUILDING

VARIANTS

- Automatic curriculum (model loss determines difficulty progression)
- Self-play curriculum (model competes with previous versions)

KEY PAPERS

[1] Textbooks Are All You Need (Phi-1) — Gunasekar et al. (Microsoft), 2023

[2] Phi-3 Technical Report: A Highly Capable Language Model — Microsoft Research, 2024

[3] Automatic Curriculum Learning for Language Model Training — Various, 2024

OPEN SOURCE

`Phi-3 training recipe (Microsoft)`

`Axolotl (https://github.com/axolotl-ai-cloud/axolotl)`

COMMON MISTAKES

- *Frontier difficulty estimated incorrectly (too easy or too hard)*
- *Not enough data at the frontier level*
- *Ignoring forgetting — need to mix in easier examples*

FUTURE DIRECTIONS

On-the-fly curriculum generation where a small orchestrator model monitors training loss and generates precisely the data the main model needs at each step — optimal learning trajectory without pre-planned curriculum.

RECIPE 07

Data Flywheels

COMPLEXITY

★★★★★

COMPUTE

Medium-High — infrastructure cost dominates; 4–32 GPUs per retraining cycle

PURPOSE

Create closed-loop systems where deployed models generate training data from real-world usage, which is filtered and used to train improved models — enabling continuous, self-sustaining improvement.

USED BY

PRODUCTION RLHF PIPELINES | SEARCH RANKING (GOOGLE, BING) | RECOMMENDATION SYSTEMS | SELF-IMPROVING AGENTS

CORE IDEA

A data flywheel connects deployment back to training: a model is deployed, serves users, logs its interactions (queries, completions, user feedback), the logged data is filtered and cleaned, and the cleaned data is used to train the next model version. The key components are data logging infrastructure, quality filters (feedback signals, raters, automated checks), and regular retraining cycles. The flywheel compounds over time — better models generate better data, which trains even better models.

TRAINING PIPELINE

ADVANTAGES

- Continuous improvement without manual data collection
- Data naturally reflects real-world usage distribution
- Improves on the cases that matter most to users

DISADVANTAGES

- Feedback signals are noisy and biased
- Quality filtering pipeline is critical infrastructure
- Can amplify biases present in user base

BEST FOR

PRODUCTION ML SYSTEMS WITH USER-FACING INTERFACES

SEARCH AND RECOMMENDATION SYSTEMS

CHATBOTS AND CONVERSATIONAL AI

VARIANTS

- Human-in-the-loop flywheel (human review before data enters training set)
- Synthetic data flywheel (model generates its own improvement data)

KEY PAPERS

[1] Scaling Data Flywheels for ML Systems — Various, 2024

[2] Learning to Improve with User Feedback — Various, 2023

[3] Continuous Improvement of LLMs through Deployment Feedback — Various, 2024

OPEN SOURCE

`LangSmith/LangFuse (logging and evaluation)`

`MLflow (experiment tracking and retraining)`

COMMON MISTAKES

- *Not filtering feedback data (training on noise degrades quality)*
- *Too long between retrain cycles (data becomes stale)*
- *Not A/B testing new model versions carefully*

FUTURE DIRECTIONS

Fully autonomous data flywheels where the model identifies its own failure modes in deployment, generates targeted improvement data, validates the fix in a sandbox, and deploys the improved version — all without human intervention.

BONUS

Decision Tree

What are you trying to build? Follow the path from your goal to the recipes that fit.

Need better reasoning?	× RLVR, DAPO, Process Supervision	Try: GRPO, On-Policy Distillation
Building a chatbot?	× Self-Instruct, Evol-Instruct, Constitutional Gen	Try: Preference Optimization, RLVR
Working with images?	× Vision RL, Self-Training Vision, Image Reward	Try: Multi-View Diffusion, Consistency Models
Generating 3D assets?	× Mesh Diffusion, Gaussian Splatting, Neural Fields	Try: Action Diffusion, Flow Matching
Processing speech?	× Speech RL, Voice Cloning, Multi-Speaker Distill	Try: Speech Token Models, Codec LMs
Deploying an agent?	× Tool-Use RL, Web Agents, Computer Use Models	Try: Multi-Agent RL, Interactive Learning
Limited compute?	× Preference Optimization, Distillation, SSM Training	Try: Linear Attention, Curriculum Generation
Lots of user data?	× Data Flywheels, Offline RL, Interactive Learning	Try: Recursive Self-Improvement, RLVR

Start → What are you building? → Follow the branch → Pick your recipe

BONUS

Compute Budget Guide

GPU-hours, parallelism, and typical wall-clock time for each recipe at production scale.

RECIPE	GPU	GPUS	DAYS	COST
GRPO	8–64	8	3–10	Medium
DAPO	8–64	8–16	3–10	Medium
On-Policy Distillation	4–32	4–8	2–7	Low-Med
RLVR	4–32	4–8	2–7	Low-Med
Preference Optimization	1–8	1–4	1–3	Low
Constitutional AI	1–8	1–4	1–3	Low
Process Supervision	4–32	4–8	2–7	Low-Med
Self-Instruct	1–8	1–4	2–5	Low-Med
Evol-Instruct	1–8	1–4	2–5	Low-Med
Judge Models	4–16	4–8	2–5	Medium
Vision RL	8–64	8–16	3–10	Medium
Multi-View Diffusion	8–64	8–16	5–14	Medium
Mesh Diffusion	4–32	4–8	3–10	Low-Med
Neural Field Training	4–16	4–8	2–7	Low-Med
Flow Matching	4–32	4–8	3–10	Low-Med
Rectified Flow	4–32	4–8	3–10	Low-Med
Gaussian Splatting	1–8	1–4	1–3	Low
Consistency Models	4–32	4–8	2–7	Low-Med
Speech RL	8–64	8–16	5–14	High
Speech Token Models	4–32	4–8	3–10	Medium
Voice Cloning	1–8	1–4	1–5	Low-Med
Codec Language Models	8–64	8–16	5–14	High
Sim-to-Real	1–8	1–4	3–10	Low-Med
Behavior Cloning	1–8	1–4	1–3	Low
Offline RL	4–32	4–8	2–7	Low-Med
World Models (Dreamer)	8–64	8–16	5–14	High
Tool-Use RL	4–32	4–8	3–10	Medium
Multi-Agent RL	8–128	16–32	7–21	Very High
Interactive Learning	4–32	4–8	3–10	Medium
Data Flywheels	4–32	4–16	ongoing	Med-High
SSM Training (Mamba)	8–64	8–16	5–14	Medium
Linear Attention	4–32	4–8	3–10	Medium

GPU counts assume H100 (80 GB). Multiply by 2–3× for A100. Costs vary by region and spot availability.

BONUS

Dependency Map

Which recipes build on which. Prerequisites left, dependents right.

GRPO	→	RLVR, DAPO
Process Supervision	→	RLVR, GRPO
Preference Optimization	→	Constitutional AI, Diffusion Pref Opt
On-Policy Distillation	→	Skill Distillation, Multi-Speaker Distillation
Self-Instruct	→	Evol-Instruct, Curriculum Generation, Synthetic Curriculum
Flow Matching	→	Rectified Flow, Action Diffusion
Multi-View Diffusion	→	Mesh Diffusion, 3D Generation pipeline
Neural Field Training	→	Gaussian Splatting Supervision
Speech Token Models	→	Codec Language Models, Voice Cloning
Codec Language Models	→	Speech RL, Multi-Speaker Distillation
Behavior Cloning	→	Offline RL, Interactive Learning
World Models (Dreamer)	→	Latent Planning, Scene-Graph Planning
Offline RL	→	Multi-Agent RL, Tool-Use RL
Tool-Use RL	→	Web Agents, Computer Use Models
Interactive Learning	→	Recursive Self-Improvement
Data Flywheels	→	Recursive Self-Improvement
Linear Attention	→	SSM Training (Mamba), State-Space Models
Synthetic Curriculum	→	Curriculum Generation
Self-Training Vision	→	Vision RL, Image Reward Models

BONUS

Timeline of Training Paradigms

How training paradigms evolved from 2014 to present, and where we are heading.

2014–2017

The Foundations

Attention is All You Need (Transformer) · Seq2Seq + Attention · Teacher Forcing · Curriculum Learning

2017–2020

Pre-Training Era

GPT / BERT pre-training · Masked Language Modeling · Next Sentence Prediction · Unsupervised Pre-training

2020–2022

Instruction Tuning

FLAN / T0 instruction tuning · Self-Instruct · RLHF (InstructGPT) · Constitutional AI

2022–2023

Alignment & Preferences

Preference Optimization (DPO) · Process Reward Models · Constitutional AI · GRPO (Group Relative PO)

2023–2024

Scaling Synthesis

Evol-Instruct & WizardLM · Synthetic Data Pipelines · Curriculum Generation (Phi) · Diffusion for 3D/Speech

2024–2025

Agentic & Self-Improving

Tool-Use RL / Web Agents · Computer Use Models · Multi-Agent RL · Recursive Self-Improvement · Data Flywheels

2025–2026

The Frontier

On-Policy Distillation at Scale · DAPO & Decoupled RL · World Models for Planning · Autonomous Curriculum Evolution