

Master's Project Report

Wenquan Lu

Included Papers

PrAg-PO: Prompt Augmented Policy Optimization for Robust and Diverse Mathematical Reasoning¹

Wenquan Lu, Hai Huang, Enqi Liu, Randall Balestrierio
Preprint.

Ditch the Denoiser: Emergence of Noise Robustness in Self-Supervised Learning from Data Curriculum¹

Wenquan Lu, Jiaqi Zhang, Hugues Van Assel, Randall Balestrierio
Published at NeurIPS 2025.

Submitted in fulfillment of the requirements for the Master's project

May 8, 2026

¹Paper approved as a Master's report for this author.

PrAg-PO: Prompt Augmented Policy Optimization for Robust and Diverse Mathematical Reasoning

Wenquan Lu¹

Hai Huang²

Enqi Liu³

Randall Balestriero¹

¹Brown University ²Palona AI ³Harvard University
{wenquan_lu, randall_balestriero}@brown.edu
hai@palona.ai enqi_liu@hsph.harvard.edu

Abstract

Reinforcement learning algorithms such as group-relative policy optimization (GRPO) have shown strong potential for improving the mathematical reasoning capabilities of large language models. While a growing body of work seeks to improve training entropy, rollout diversity, and exploration, most existing methods still train models with a single fixed reasoning prompt or template, which can encourage prompt-specific overfitting and unstable training dynamics. In this work, we introduce Prompt Augmented Policy Optimization (PrAg-PO), a simple policy optimization method that mixes prompt templates with template-specific format rewards during training. By encouraging models to generate reasoning traces under diverse instructions and output formats, PrAg-PO increases rollout diversity and improves robustness. Compared with GRPO and DAPO, PrAg-PO achieves significantly higher reasoning accuracy while mitigating premature training collapse. Empirically, experiments on DeepSeek-R1-Distill-Qwen-1.5B, Qwen2.5-Math-1.5B, and Qwen3-1.7B show that PrAg-PO consistently outperforms strong baselines and achieves competitive performance against recent methods on mathematics benchmarks, using only a fixed MATH Level 3-5 training set of 8.5K problems. The code and model checkpoints are available at <https://github.com/wenquanlu/PrAg-PO>.

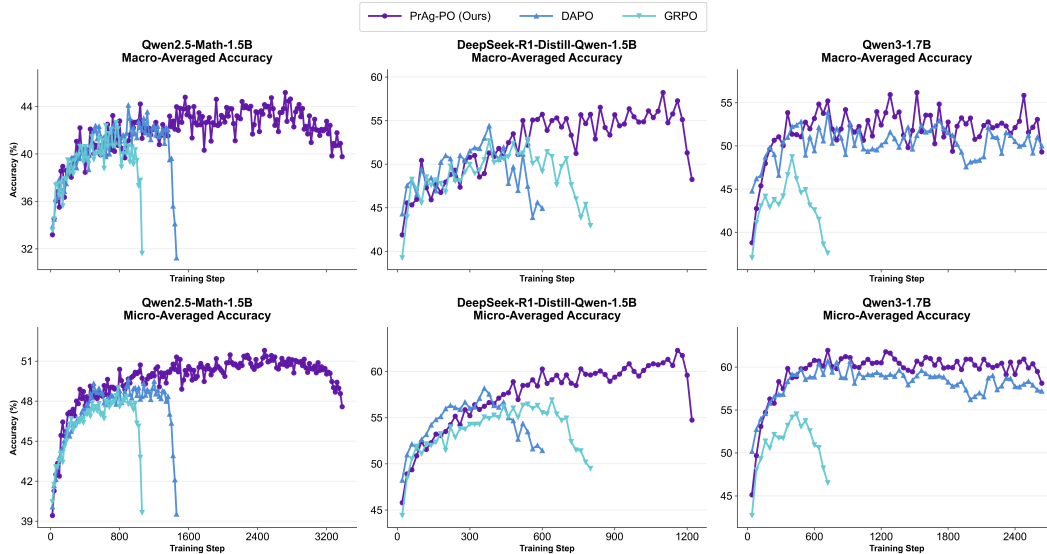


Figure 1: Evaluation accuracy over training steps averaged over mathematics benchmarks for models trained on MATH Level 3-5. PrAg-PO significantly outperforms GRPO and DAPO across Qwen2.5, DeepSeek-R1-Distilled and Qwen3 models, substantially delaying the overfitting and model collapse.

1 Introduction

Since the introduction of GRPO [41] and DeepSeek-R1 [10], which are built upon classical reinforcement learning algorithms such as TRPO [39] and PPO [40], reinforcement learning (RL) has demonstrated strong capability in improving models’ reasoning performance on verifiable tasks, including mathematics [55, 56, 59, 58, 8], coding [36], medical diagnosis [47], and tabular reasoning [54]. RL has proven to be an effective post-training paradigm for large language models.

Numerous empirical and theoretical efforts have been made to improve the training objective of the GRPO algorithm. Notably, DAPO [55], Dr. GRPO [29], and GSPO [59] unanimously remove the KL loss term from the original GRPO objective, arguing that it may limit reasoning performance by constraining the trained model to remain too close to the reference model. In addition, many works [5, 4, 42] have proposed a variety of entropy regularization techniques to address entropy collapse phenomenon: a sharp decrease in training entropy as reinforcement learning post-training progresses. Entropy serves as a proxy for policy stochasticity and output diversity. Viewing an LLM as a softmax policy over tokens, entropy collapse corresponds to probability mass concentrating on a narrow subset of tokens, severely limiting the diversity of sampled reasoning trajectories.

Despite the advancements to promote diversity, almost all existing work trains LLMs using a single reasoning format per training run, with a fixed set of format rewards. For example, DeepSeek-R1 [10] and Open-R1 [14] adopt a tagged reasoning format (e.g., <think>...</think><answer>...</answer>), while SimpleRL-Zoo [56] and OAT-Zero [28] rely on Qwen’s original chain-of-thought prompting with free-form generation. We hypothesize that such homogeneous reasoning formats encourage overfitting to a single reasoning style, thereby reducing reasoning diversity. Moreover, given the expressive power of chain-of-thought reasoning, there exists a large and largely unexplored design space in which models can reason in fundamentally different ways. Also, given LLM’s notorious sensitivity to prompts [53, 31], optimizing across different prompt formats can improve model’s robustness to prompt variations, thereby potentially improve generalization and test-time performance.

In this work, we introduce PrAg-PO, a simple yet effective policy optimization technique for RL post-training of LLMs on mathematical reasoning. Specifically, we mix multiple reasoning templates and formats within a single training run, including tagged reasoning–answer separation, free-form generation, explicit chain-of-thought prompting, and reflection-based formats. These templates are paired with template-specific format rewards to ensure faithful adherence during training. We show that PrAg-PO successfully elicits diverse reasoning behaviors within a single model trained in a single run. Importantly, under a fixed training dataset and the default evaluation prompt template, PrAg-PO achieves competitive performance on mathematical reasoning benchmarks. As shown in Figure 1, our method outperforms both vanilla GRPO and DAPO in terms of per-benchmark mean accuracy and per-question mean accuracy on Qwen2.5-Math-1.5B, DeepSeek-R1-Distill-Qwen-1.5B and Qwen3-1.7B. It is important to note that such improvement is non-trivial, as mixed template training inherently causes distributional difference between train and test time, potentially impairing the performance. We also find that prompt augmentation helps stabilizing RL training, reducing overfitting and allowing continued policy improvement. In summary, our contributions are three-fold:

1. We propose Prompt Augmented Policy Optimization (PrAg-PO) for RL-based mathematical reasoning, a simple, effective, and computationally inexpensive policy optimization method that employs a diverse set of reasoning templates with associated format rewards to train models to reason. We show that this approach elicits a richer set of reasoning trajectories.
2. Empirically, prompt augmentation stabilizes training dynamics and substantially delays model collapse, enabling longer training horizons. We further interpret prompt augmentation as a gradient regularizer that smooths optimization and reduces overfitting to a single prompt-reward interface.
3. Across different model families, PrAg-PO significantly outperforms GRPO and DAPO, achieving competitive performance that exceeds leading results from prior literature on standard mathematical reasoning benchmarks, as shown in Table 1, Table 2, and Table 3.

2 Related Works

Reinforcement Learning with Verifiable Rewards. DeepSeek-R1 shows that rule-based verifiable rewards using GRPO algorithm can significantly improve the reasoning capability of pre-trained

LLMs. This leads to a surge in a new research paradigm. DAPO [55] proposes decoupled-clip, dynamic sampling, dropping KL penalty and token-level gradient loss that promotes exploration diversity. Dr. GRPO [29] identifies a length bias in original GRPO’s objective and proposes removing normalization term. GSPO [59] proposed sequence-level importance ratios which notably stabilize RL training for MoE models. GMPO [58] optimizes geometric-mean instead of the arithmetic-mean of token-level rewards. MO-GRPO [15] and GDPO [27] proposes to decouple the normalization of individual rewards in multi-reward settings. Overall, most research centers on enhancing the stability and performance of RLVR algorithms.

Increasing Data and Rollout Diversity for RL. Early seminal works in RL [49] have added an entropy regularization term to the policy objective to encourage exploration [34, 40, 11]. However, it does not result in notable performance gains for LLMs [5] due to LLM’s extremely large response space [42], which has led to a growing body of work on entropy regularization [5, 4, 42]. Notably, DRA-GRPO [3] proposes a diversity-aware reward to increase rollout diversity by leveraging sub-modular mutual information. Beyond-Pass@1 [24] shows that problem synthesis and self-play can maintain policy entropy during training. Recent or concurrent works [21, 6, 19] also propose question augmentation, which augments question content via partial answers or reformulations. However, these methods do not focus on augmenting prompt templates. Moreover, question-level augmentation is typically more expensive and requires carefully designed curricula. In contrast, our method is substantially lighter and easier to reproduce.

LLM’s Notorious Sensitivity to Prompts. LLM’s performance has been shown to be notoriously sensitive to the prompts. Chain-of-Thought (CoT) prompting has been shown to substantially improve LLM reasoning performance [48, 30]. [48] demonstrate few-shot CoT demonstrations enable complex reasoning, while [16] show that LLMs can perform zero-shot reasoning by simply prepending “Let’s think step by step” to the prompt. DeepSeek-R1 [10] adopts zero-shot prompting and explicitly reports that few-shot prompting consistently degrades performance, indicating that RL-trained models are highly sensitive to evaluation prompts. In contrast, Dr. GRPO [29] shows that RL training itself is sensitive to the choice of prompt templates. Large language models as optimizer [53] show that one can improve LLM’s performance on mathematical reasoning simply by optimizing test-time prompts.

3 PrAg-PO: Prompt Augmented Policy Optimization

3.1 Preliminary: Group-Relative Policy Optimization

Group Relative Policy Optimization (GRPO) is a variant of proximal policy optimization (PPO) algorithm that replaces value-function baselines with group-relative normalization. For each question q drawn from a training distribution, GRPO samples a group of rollouts $\{o_1, o_2, \dots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, and maximizes the following surrogate objective:

$$\mathcal{L}_{\text{GRPO}_{\text{token}}}(\theta) = \mathbb{E}_{q \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \right. \quad (1)$$

$$\left. \left(\min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right) - \beta D_{KL}^{i,t} \right) \right], \quad (2)$$

Where

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}$$

Instead of using sample-level loss proposed in the original GRPO formulation, we here express token-level policy-gradient loss, which is widely adopted in most RL libraries (e.g., TRL [46], verl [43] and slime [60]), leading to fairer credit assignment especially in long-CoT scenarios.

3.2 Prompt Augmentation

Here we introduce the formulation of prompt augmented policy optimization. Given a set of template functions $\{T_1, T_2, \dots, T_K\}$ and corresponding reward functions $\{R_1, R_2, \dots, R_K\}$, each template

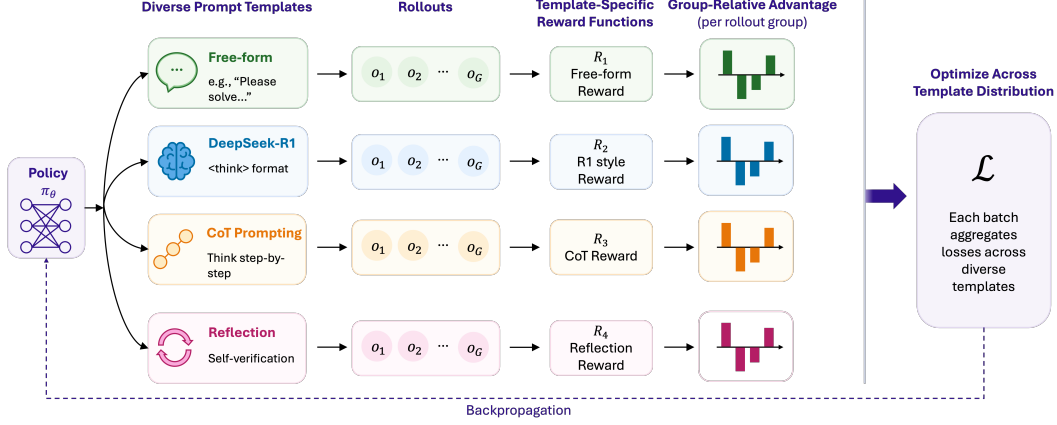


Figure 2: Outline of Prompt Augmented Policy Optimization. The policy is conditioned on diverse prompt templates to generate grouped rollouts. For each group, a template-specific reward function is applied and group-relative advantages are computed. Each batch update therefore aggregates losses across diverse prompt templates.

function contains unique prompts and instructions that instruct the model to reason in a certain format. At training time, for each question q drawn from a training distribution, we also uniformly sample a template function and apply it to the question q . We thus maximize the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{q \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}, k \sim \text{Uniform}(\{1, \dots, K\})} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \right] \quad (3)$$

$$\sum_{t=1}^{|o_i|} \min \left(r_{i,t}^k(\theta) \hat{A}_{i,t}^k, \text{clip}(r_{i,t}^k(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}) \hat{A}_{i,t}^k \right), \quad (4)$$

where

$$r_{i,t}^k(\theta) = \frac{\pi_{\theta}(o_{i,t} | T_k(q), o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | T_k(q), o_{i,<t})}, \quad \hat{A}_{i,t}^k = \frac{R_k(o_i) - \text{mean}(\{R_k(o_i)\}_{i=1}^G)}{\text{std}(\{R_k(o_i)\}_{i=1}^G)}$$

Note that we drop the KL penalty as a common practice [55, 29], and use decoupled clipping for entropy regularization. Using the above objective with prompt augmentation, we can elicit the language models to reason in diverse formats in a single training run. Moreover, each group of rollouts has the same template, which allows valid computation of advantage. As each batch contains multiple groups, each update still incorporates gradients from different templates, as shown in Figure 2.

Optimization View: Prompt Augmentation as Gradient Regularizer. From an optimization perspective, prompt augmentation can be viewed as optimizing a prompt-perturbed training objective that regularizes policy optimization under the fixed training question distribution. Let T_k denote a prompt template sampled from $P(T)$, where $P = \text{Uniform}(\{1, \dots, K\})$. For each template, define the template-conditioned population gradient as

$$G_k(\theta) = \mathbb{E}_{q \sim \mathcal{D}} [\nabla_{\theta} \mathcal{L}_{T_k}(q; \theta)].$$

PrAg-PO optimizes the augmented objective whose population gradient is

$$G_{\text{aug}}(\theta) = \mathbb{E}_{k \sim P(T)} [G_k(\theta)].$$

Although evaluation is conducted under a single default prompt template, optimizing the augmented objective can still improve the default-template policy by acting as a regularizer during training. To see this, decompose the template-conditioned population gradient as

$$G_k(\theta) = G_{\text{shared}}(\theta) + \xi_{\text{temp}}(k; \theta),$$

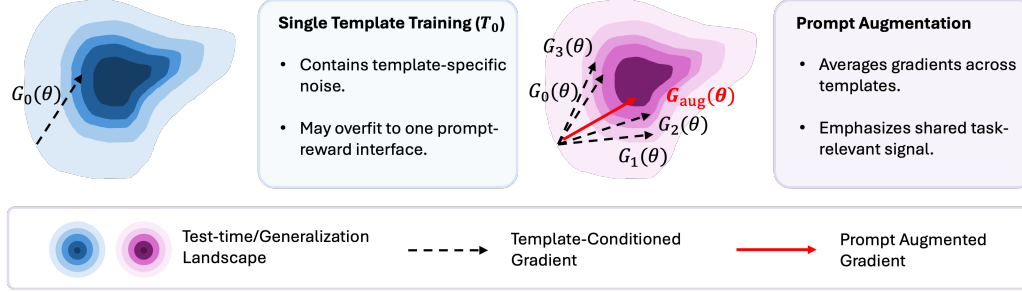


Figure 3: **Prompt augmentation as a gradient regularizer.** Left: Single-template training follows a template-conditioned gradient $G_0(\theta)$, which includes template-specific noise and can lead to overfitting to one prompt-reward interface. Right: Prompt augmentation samples multiple templates and averages their gradients $G_{\text{aug}}(\theta) = \mathbb{E}_{k \sim P(T)}[G_k(\theta)]$, attenuating template-specific components while preserving shared task-relevant signals, resulting in improved generalization.

where $G_{\text{shared}}(\theta)$ denotes gradient components that consistently improve mathematical reasoning across prompt-reward interfaces, while $\xi_{\text{temp}}(k; \theta)$ captures template-specific components induced by prompt wording, output structure, and format-reward design.

As illustrated in Figure 3, a single-template method repeatedly follows $G_0(\theta)$, and may therefore over-optimize template-specific components associated with the default training prompt. Such components can reduce stability and lead to brittle reasoning behavior. In contrast, PrAg-PO averages updates over multiple prompt-reward interfaces. Under the idealized case where template-specific components are zero-mean across the prompt distribution, $\mathbb{E}_{k \sim P(T)}[\xi_{\text{temp}}(k; \theta)] \approx 0$, the augmented gradient reduces to the shared reasoning gradient, $G_{\text{aug}}(\theta) \approx G_{\text{shared}}(\theta)$. In practice, this cancellation need not be exact. As long as template-specific components are less consistently aligned across templates than the shared reasoning components, averaging over prompt-reward interfaces attenuates prompt-idiosyncratic directions relative to a single-template update. Thus, the prompt-augmented objective acts as a training-time regularizer, promoting more robust and generalizable gradients. This is consistent with our empirical results in Figure 1, where PrAg-PO delays training collapse, and with the entropy analysis in Appendix A.2 where it exhibits smoother token-level policy entropy.

3.3 Prompt Template Curation and Selection

We curate a diverse range of reasoning templates and formats to encourage the model to reason differently. Appendix B illustrates examples of templates. Generally, the templates can be divided into four categories. The first group is the DeepSeek Style template that enforces the model to put reasoning processes within special tags like <think> tags and <answer> tags. Such templates are widely used in current literature and open-source projects [14, 54]. The second group is the free-form reasoning generation, which is Qwen’s default style template. The model is instructed to output reasoning processes, and no format constraints are imposed except the final answer format. This style of template is also widely adopted in Qwen native training scenarios [56, 29]. The third group is reflection-based template. The model is instructed to check its work, identify any potential error, and put the verification processes within <check> tags. This form of reasoning is relatively underexplored in current literature, as most work investigated the natural emergence of "aha", "wait" or reflection moments rather than enforce an explicit format for reflection. The fourth group conditions the model on explicit chain-of-thought prompting. The models are conditioned on phrases like "Let’s think step by step", and then continue the generation. For some templates with special tags, we additionally include a teacher-forced variant, where the assistant generation is initialized with the corresponding reasoning tags (e.g., <think> or <solution>). This provides stronger structural guidance during training and improves convergence speed.

Initially, we collect 13 prompt templates from open-source models and evaluation frameworks, including Qwen [52], DeepSeek-R1 [10], Open-R1 [14], and lm-evaluation-harness [9]. We use these templates to train Qwen2.5-Math-1.5B and Qwen3-1.7B. In addition, we find that DeepSeek-R1-Distill-Qwen-1.5B benefits substantially from greater template diversity. Therefore, inspired by Toolformer [38], we leverage the in-context learning capability of a proprietary LLM [35] to expand the template pool to 35 prompt templates for training the distilled model, as detailed in Appendix B.3

Table 1: Evaluation results on mathematics benchmarks for methods fine-tuned from Qwen2.5-Math-1.5B with reinforcement learning. See Appendix Table 4 for evaluation standard deviation.

Method	AIME24	AMC(23&24)	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
Qwen2.5-Math-1.5B	16.7	43.4	61.8	15.1	28.4	33.1	37.4
GRPO	23.3	49.9	75.4	26.3	38.3	42.6	48.4
Dr. GRPO	20.0	53.0	74.2	25.7	37.6	42.1	47.7
SEED GRPO	23.3	50.6	75.4	26.8	41.3	43.5	49.8
GMPO	20.0	53.0	77.6	30.1	38.7	43.9	50.1
DAPO	23.3	54.9	76.9	26.0	39.4	44.1	49.6
PrAg-PO (Step 2720)	23.3	53.5	78.4	31.3	39.4	45.2	50.9
PrAg-PO (Step 2480)	20.0	49.4	80.4	29.7	41.2	44.1	51.8

Table 2: Evaluation results on mathematics benchmarks for methods fine-tuned from DeepSeek-R1-Distill-Qwen-1.5B with reinforcement learning. See Appendix Table 5 for standard deviation.

Method	AIME24	AMC23	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
DeepSeek-R1-Distill-Qwen-1.5B	28.8	62.9	82.8	26.5	43.3	48.9	53.5
Still-3-1.5B-Preview	32.5	66.7	84.4	29.0	45.4	51.6	55.6
DeepScaleR-1.5B-Preview	43.1	73.6	87.8	30.2	50.0	57.0	59.4
Open-RS1	30.0	70.0	83.8	29.0	52.4	53.0	58.6
Open-RS2	30.0	80.0	85.4	30.5	52.4	55.7	59.6
Open-RS3	46.7	72.5	84.4	26.8	51.3	56.3	58.3
Dr. GRPO	33.3	80.0	83.4	30.5	52.1	56.0	58.9
DRA-GRPO	36.7	75.0	86.2	32.4	53.0	56.7	60.5
DRA-Dr.GRPO	36.7	85.0	85.2	30.5	53.8	58.2	60.5
GRPO	30.0	74.0	85.5	28.2	46.5	52.8	56.5
DAPO	33.3	77.5	84.2	26.9	50.0	54.4	57.6
PrAg-PO (Step 1100)	42.0	78.5	87.5	29.3	53.8	58.2	60.9
PrAg-PO (Step 1160)	33.3	77.0	87.2	32.6	56.2	57.3	62.3

Table 3: Evaluation results on mathematics benchmarks for methods fine-tuned from Qwen3-1.7B with reinforcement learning. See Appendix Table 6 for evaluation standard deviation.

Method	AIME24	AMC23	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
Qwen3-1.7B	3.3	27.5	57.9	19.1	18.2	25.2	32.9
GAC	20.0	54.1	73.6	25.1	28.1	40.2	43.1
RePO	12.3	47.7	59.6	20.2	63.1	40.6	52.8
GRPO	16.7	70.0	78.7	33.2	45.1	48.7	54.1
DAPO	26.7	70.0	84.9	32.8	55.1	53.9	60.7
PrAg-PO (Step 1520)	33.3	75.0	85.1	33.6	53.9	56.2	60.7
PrAg-PO (Step 720)	23.3	76.5	87.0	33.5	55.7	55.2	62.0

3.4 Template-Specific Reward Functions

The reward for PrAg-PO is composed of an accuracy reward and a format reward, both ranging from 0 to 1. We use binary rewards for accuracy, and follow TreePO [23], using Math-Verify [18] and SymPy [33] libraries to verify a response by extracting the final answer enclosed within `\boxed{}`.

Template-Specific reward functions are critical to elicit diverse reasoning formats from the model. We will show in ablation studies that without these specific reward functions, the model will simply ignore the instruction and still collapse prematurely. Nevertheless, we found that simple string matching and tag-count-based rewards are sufficient to induce the desired reasoning format, without requiring complex regular expressions (Regex) matching. As shown in Appendix B for DeepSeek-style format, format rewards are computed based on the exact match of special tags. Partial rewards are assigned when the model generates a subset of the required tags; missing or duplicated

tags receive no reward. Moreover, the ordering of the tags are not examined, as empirically the model can naturally develop correct ordering. The observation that language models are quick format learners are also observed in other literature [50], however, we are the first to formally explore it in a hybrid format setting. For templates that do not impose format constraints, we assign the format reward a constant value of 1 to ensure a consistent scale across templates. Since the GRPO objective normalizes rewards within each group, this constant has no effect on the advantage estimates.

4 Experiments

4.1 Experimental Settings

Datasets. We use the MATH Level 3-5 dataset [56] as the training set, which is a subset of 8,523 questions derived from the MATH dataset [13]. This filtered subset contains problems of medium to high difficulty (i.e., levels 3 to 5 on a five-level scale). The MATH dataset consists of problems from mathematics competitions, including AMC 10, AMC 12, and AIME. Our test set comprises a standard suite of benchmarks, including AIME24 (30 questions) [57], AMC (83 questions) [1], AMC23 (40 questions), MATH500 (500 questions), Minerva Math (272 questions) [20], and OlympiadBench (675 questions) [12], covering a broad range of graduate-level and Olympiad-style problems.

Models and Competitors. We use Qwen2.5-Math-1.5B [52], DeepSeek-R1-Distill-Qwen-1.5B [10], and Qwen3-1.7B [44] as base models, covering different model families and training stages. Qwen2.5-Math-1.5B is continually pretrained from Qwen2.5-1.5B on mathematical corpora. DeepSeek-R1-Distill-Qwen-1.5B is distilled from the large DeepSeek-R1 model. Qwen3-1.7B is a latest-generation instruction-tuned model.

We reproduce GRPO [41] and DAPO [55] baselines using our code base. For GRPO, we adopt token-level loss. KL coefficient β is 0.001, consistent with DeepSeek-R1. For DAPO, we implement decoupled clip and KL penalty removal, following the core training objective. In addition to GRPO and DAPO, we compare our method against a comprehensive set of state-of-the-art baselines. For Qwen2.5-Math-1.5B, we include Dr. GRPO [29], SEED-GRPO [2], and GMPO [58]. For DeepSeek-R1-Distill-Qwen-1.5B, we include Still-3-1.5B-Preview [45], DeepScaleR-1.5B-Preview [32], OpenRS [7], Dr. GRPO [29], and DRA-GRPO [3]. For Qwen3-1.7B, we include GAC [51] and RePO [22].

Training and Evaluation Settings. We train our models using the verl framework [43]. We use a prompt batch size of 128 with a mini-batch size of 32, resulting in four weight updates per batch. The group size is set to 8, so each mini-batch contains 256 rollouts. Following DAPO, we set the clipping parameters to $\epsilon_{\text{high}} = 0.28$ and $\epsilon_{\text{low}} = 0.20$. For Qwen2.5-Math-1.5B and Qwen3-1.7B, we cap the maximum prompt length at 1024 tokens and the maximum output length at 3072 tokens during training, and use a maximum model length of 4096 tokens during evaluation. For DeepSeek-R1-Distill-Qwen-1.5B, following DRA-GRPO [3], we cap the maximum prompt length at 1024 tokens and the maximum output length at 3584 tokens during training, and use a maximum model length of 32768 tokens during evaluation. We use AdamW optimizer and a constant learning rate of 1×10^{-6} . All training runs are conducted on 8 L40S GPUs, with vLLM serving as the inference engine. Each training run takes approximately 1.5 - 4 days to complete depending on the total optimization steps.

To ensure fair comparison and avoid introducing test-time scaling effects, **we use model tokenizer’s single default template during evaluation**, adopting exactly the same evaluation framework and code base as Dr. GRPO, SEED GRPO and GMPO. We evaluate models’ performance under greedy decoding (i.e., temperature = 0), and report pass@1 metric. Furthermore, due to inherent stochasticity of vLLM inference engine [17], the model may produce different responses even under the same seed. To ensure reproducibility and reliability of our results, we let all models perform five inference rounds under the same seed, and report the average accuracy. Importantly, due to the extreme imbalanced nature of the test datasets, we report both per-benchmark average accuracy and per-question average accuracy. Reporting both metrics avoids bias toward either large benchmarks or small benchmarks in a highly imbalanced test suite. Following standard practice in Dr. GRPO and DAPO, the model is evaluated at periodic checkpoints, and the best-performing results are reported in the table.

4.2 PrAg-PO Improves Reasoning Accuracy and Stabilizes Group-Relative Policy Training

PrAg-PO demonstrates strong performance in both internal comparisons against our GRPO and DAPO baselines and external comparisons against previously reported methods. As shown in Table 1, Table 2, and Table 3, PrAg-PO consistently outperforms the GRPO and DAPO baselines in both

Let $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, where $i = \sqrt{-1}$. What is $f(2022)$?		
Qwen-Math Template Output	DeepSeek-Style Template Output	Reflection-Based Template Output
Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we need to determine the value of $f(2022)$. First, let's denote the complex numbers $\omega_1 = \frac{-1+i\sqrt{3}}{2}$ and $\omega_2 = \frac{-1-i\sqrt{3}}{2}$. These are the cube roots of unity, specifically $\omega_1 = \omega$ and $\omega_2 = \omega^2$, where $\omega = e^{2\pi i/3}$ and $\omega^2 = e^{-2\pi i/3}$. This means $\omega^3 = 1$ and $(\omega^2)^3 = 1$. Thus, we have $\omega_1^3 = 1$ and $\omega_2^3 = 1$. Therefore, $\omega_1^{2022} = (\omega_1^3)^{674} = 1^{674} = 1$ and $\omega_2^{2022} = (\omega_2^3)^{674} = 1^{674} = 1$. Hence, $f(2022) = \omega_1^{2022} + \omega_2^{2022} = 1 + 1 = 2$. So, the value of $f(2022)$ is 2. The final answer is $\boxed{2}$.	<think> Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we can denote the complex numbers $\omega = \frac{-1+i\sqrt{3}}{2}$ and $\omega^2 = \frac{-1-i\sqrt{3}}{2}$. These are the non-real cube roots of unity, satisfying $\omega^3 = 1$ and $(\omega^2)^3 = 1$, and also $\omega^2 + \omega + 1 = 0$. Thus, we have $f(n) = \omega^n + (\omega^2)^n$. We need to find $f(2022)$. Since $\omega^3 = 1$, we can reduce the exponent modulo 3: $2022 \bmod 3 = 0.$ Therefore, $\omega^{2022} = (\omega^3)^{674} = 1^{674} = 1$ and $(\omega^2)^{2022} = (\omega^2)^{3 \cdot 674} = (\omega^3)^{674} = 1^{674} = 1$. Hence, $f(2022) = \omega^{2022} + (\omega^2)^{2022} = 1 + 1 = 2$. The value of $f(2022)$ is 2. </think> <answer> The value of $f(2022)$ is $\boxed{2}$. </answer>	<solution> Given the function $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \left(\frac{-1-i\sqrt{3}}{2}\right)^n$, we can denote the complex numbers $\omega = \frac{-1+i\sqrt{3}}{2}$ and $\omega^2 = \frac{-1-i\sqrt{3}}{2}$. These numbers are the non-real cube roots of unity, and they satisfy the property $\omega^3 = 1$ and $\omega^2 = 1$. We need to find $f(2022)$. First, we observe the periodicity of ω and ω^2: $\omega^3 = 1 \implies \omega^{2022} = (\omega^3)^{674} = 1^{674} = 1$ $\omega^2 = 1 \implies \omega^{2022} = (\omega^2)^{674} = 1^{674} = 1$ Thus, we have: $f(2022) = \omega^{2022} + \omega^{2022} = 1 + 1 = 2$ Therefore, the value of $f(2022)$ is 2. The final answer is $\boxed{2}$. </solution> <check> Let's verify step by step: 1. We calculated that $\omega^{2022} = 1$ and $\omega^{2022} = 1$. 2. Therefore, $f(2022) = \omega^{2022} + \omega^{2022} = 1 + 1 = 2$. The calculation is correct. The value of $f(2022)$ is indeed 2. </check>

Figure 4: Reasoning trajectories produced by the model trained with PrAg-PO when answering the same question under different prompt templates. The model generates diverse reasoning formats.

average benchmark accuracy and average question accuracy, indicating that its performance gains are robust across evaluation criteria. Compared with other competitive and state-of-the-art methods, PrAg-PO also achieves leading performance across model families. For Qwen2.5-Math-1.5B, shown in Table 1, PrAg-PO achieves the best performance under both evaluation metrics. Notably at step 2480, to the best of our knowledge, this is the first instance where a Qwen2.5-Math-1.5B RL-finetuned model surpasses 80% accuracy on the MATH500 benchmark. For DeepSeek-R1-Distill-Qwen-1.5B, shown in Table 2, PrAg-PO again achieves the highest average accuracy in both metrics, having a 2-point lead in per-question accuracy, while also obtaining the best score on OlympiadBench and Minerva. For Qwen3-1.7B, shown in Table 3, PrAg-PO substantially outperforms all competing methods, exceeding the strongest previously reported result (i.e., RePO) by more than 10 points. These results demonstrate that prompt augmentation is an effective and broadly applicable method for improving the reasoning accuracy of language models under reinforcement learning fine-tuning.

Training collapse during prolonged GRPO training has been observed in numerous prior studies [7, 37, 25, 26]. Although its exact causes remain an active area of research, recent work has attributed such instability to factors such as entropy collapse and training-inference precision mismatch. Empirically, we find that PrAg-PO is effective in stabilizing training dynamics and delaying collapse. As shown in Figure 1, Dr. GRPO and DAPO collapse after roughly 400–1500 steps. In contrast, PrAg-PO continues to improve over a substantially longer training horizon, reaching more than 2000 steps. This effect is also reflected in token-level policy entropy, where PrAg-PO exhibits smoother and more stable entropy trajectories compared to DAPO (See Appendix A.2). This improved stability may be driven by the regularizing effect of prompt augmentation. By exposing the policy to diverse prompt and reward interfaces, PrAg-PO reduces overfitting to any single reasoning template. Moreover, aggregating losses across different templates smooths the overall gradient signal, as illustrated in Section 3.2, attenuating template-specific noise and reducing abrupt spikes in training dynamics.

4.3 PrAg-PO Elicits Diverse Reasoning Trajectories

In this section, we demonstrate that PrAg-PO enables the model to develop diverse reasoning formats and trajectories. As shown in Figure 4, we evaluate a single trained checkpoint on the same mathematical question, “Let $f(n) = \left(\frac{-1+i\sqrt{3}}{2}\right)^n + \dots$ ”, while varying only the prompt template. Despite identical inputs and model parameters, the model adapts its output to follow the instructions imposed by each template, producing distinct reasoning structures. In the left column, the Qwen-Math free-form generation template is applied, resulting in a generic, unconstrained reasoning trace. In the middle column, a DeepSeek-style template instructs the model to place its reasoning within **<think>** tags and its final answer within **<answer>** tags. The model successfully adheres to these structural constraints, with the required tags highlighted in blue. In the right column, a reflection-based template is used, prompting the model to include an additional verification step enclosed in **<check>** tags, which it also follows correctly. Although the three solutions are mathematically

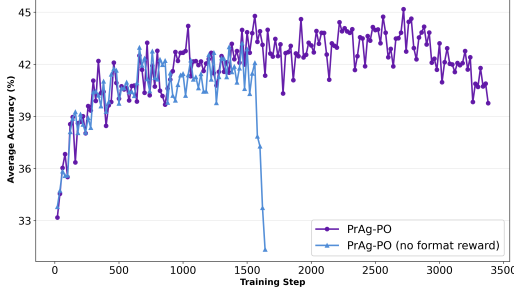


Figure 5: Accuracy of PrAg-PO with and without format rewards across training steps on Qwen2.5-Math-1.5B. Without format rewards (blue curve), the model underperforms and collapses earlier.

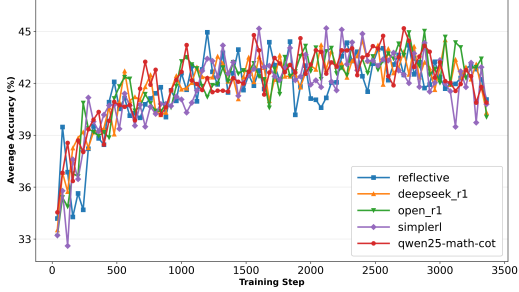


Figure 6: Accuracy of PrAg-PO evaluated with different prompt templates on Qwen2.5-Math-1.5B. Performance is consistent and robust across prompt templates, with no clear outlier.

similar, they exhibit subtle yet meaningful differences in reasoning emphasis and presentation. For instance, the left-column output explicitly converts the complex numbers into their exponential form to invoke the cube roots of unity, whereas the middle-column output emphasizes reducing the exponent 2022 modulo 3 to simplify the expression. Interestingly, the right-column output first identifies $\frac{-1-i\sqrt{3}}{2}$ as the complex conjugate of $\frac{-1+i\sqrt{3}}{2}$ before appealing to the same cube-root-of-unity property, and presents the solution in a more step-wise manner that facilitates verification. These observations illustrate that prompt augmentation elicits multiple valid reasoning trajectories from a single model, diversifying its outputs and supporting more effective GRPO training.

4.4 Ablation Studies

Template-Specific Format Rewards Are Critical for Stable Training. In this section, we examine whether template-specific format rewards are necessary for improving reasoning performance and eliciting diverse reasoning trajectories. In principle, these benefits could be attributed solely to prompt-level data augmentation, without modifying the training objective. To isolate the effect of format rewards, we remove them entirely and train the model using only a binary accuracy reward, while still applying the augmented prompt templates. The resulting accuracy curves are shown as the light blue line in Figure 5. Without format rewards, training collapses at approximately 1,500 steps, and the model consistently underperforms the counterpart trained with format rewards. Through manual inspection of the model’s rollouts, we observe that under this setting the model largely ignores the prompt instructions and defaults to free-form reasoning outputs. This behavior is expected, as in the absence of explicit reward signals, the model has no incentive to follow diverse reasoning formats.

There is No “Secret” Gold Prompt. A natural concern is that the effectiveness of PrAg-PO may be driven by the accidental inclusion of a “gold” prompt that substantially boosts training performance. In this section, we show that this is not the case: after prompt-augmented policy optimization, the model performs comparably across different prompt templates, indicating that no single template provides a disproportionate advantage. We evaluate Qwen2.5-Math-1.5B at periodic checkpoints under different prompt templates. As shown in Figure 6, accuracies are largely consistent across templates, fluctuating around a common mean without clear outliers. This suggests that the model becomes robust to prompt variation, and that the performance gains arise from the collective effect of prompt augmentation rather than from a single hidden “gold” prompt.

5 Conclusion, Limitations and Future Works

We introduce Prompt Augmented Policy Optimization (PrAg-PO) for training LLMs on mathematical reasoning tasks. We show that prompt augmentation, when combined with template-specific format rewards, elicits diverse reasoning trajectories, stabilizes training, and improves reasoning accuracy. One limitation is that PrAg-PO does not fully eliminate long-horizon optimization instability, as the model may still collapse after sufficiently prolonged training. Nevertheless, under a fixed dataset budget, PrAg-PO enables more stable training dynamics and achieves stronger performance. Another limitation is that the template pool still involves human design choices. More systematic template selection could improve scalability, although our analysis suggests that PrAg-PO does not rely on a single “gold” prompt. As a promising direction for future work, the availability of multiple reasoning formats opens the door to inference-time scaling strategies that may further enhance performance.

References

- [1] AI-MO. AIMO Validation AMC Dataset. <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>, 2024. Accessed: 2026-05-06.
- [2] Minghan Chen, Guikun Chen, Wenguan Wang, and Yi Yang. Seed-grpo: Semantic entropy enhanced grpo for uncertainty-aware policy optimization. *arXiv preprint arXiv:2505.12346*, 2025.
- [3] Xiwen Chen, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Hao Wang, Haiyu Wu, Huayu Li, Aristeidis Sotiras, Yalin Wang, and Abolfazl Razi. Dra-grpo: Exploring diversity-aware reward adjustment for r1-zero-like training of large language models. *arXiv preprint arXiv:2505.09655*, 2025.
- [4] Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.
- [5] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.
- [6] Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. Harder is better: Boosting mathematical reasoning via difficulty-aware grpo and multi-aspect question reformulation. *arXiv preprint arXiv:2601.20614*, 2026.
- [7] Quy-Anh Dang and Chris Ngo. Reinforcement learning for reasoning in small LLMs: What works and what doesn't. In *Logical and Symbolic Reasoning in Language Models @ AAAI 2026*, 2026. URL <https://openreview.net/forum?id=3pWL6Zxc4A>.
- [8] Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.
- [9] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- [10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.
- [12] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024.
- [13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
- [14] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL <https://github.com/huggingface/open-r1>

- [15] Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. Mo-grpo: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *arXiv preprint arXiv:2509.22047*, 2025.
- [16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [17] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- [18] Hynek Kydlíček. Math-Verify: Math verification library, 2024. URL <https://github.com/huggingface/math-verify>
- [19] Khiem Le, Youssef Mroueh, Phuc Nguyen, Chi-Heng Lin, Shangqian Gao, Ting Hua, and Nitesh V Chawla. Transform-augmented grpo improves pass@ k. *arXiv preprint arXiv:2601.22478*, 2026.
- [20] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.
- [21] Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. *arXiv preprint arXiv:2507.13266*, 2025.
- [22] Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025.
- [23] Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, et al. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*, 2025.
- [24] Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*, 2025.
- [25] Jiakai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Zhuo Jiang. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025. URL <https://richardli.xyz/rl-collapse>.
- [26] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=YPsJha5HXQ>
- [27] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.
- [28] Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. There may not be aha moment in rl-zero-like training — a pilot study. <https://oatllm.notion.site/oat-zero>, 2025. Notion Blog.
- [29] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding rl-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=5PAF7PAY2Y>

- [30] Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.
- [31] Wenquan Lu, Jiaqi Zhang, Hugues Van Assel, and Randall Balestriero. Ditch the denoiser: Emergence of noise robustness in self-supervised learning from data curriculum. In *Advances in Neural Information Processing Systems*, volume 38, pages 136057–136089, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c6a4fc50b10ec1add5086e2f527ff0d3-Paper-Conference.pdf
- [32] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. [Notion Blog](#), 2025.
- [33] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL <https://doi.org/10.7717/peerj-cs.103>
- [34] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.
- [35] OpenAI. Chatgpt. <https://chat.openai.com/>, 2026. Accessed: 2026-03-20.
- [36] Mohammadreza Pourreza, Shayan Taleai, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan O Arik. Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=HbwkIDWQgN>
- [37] Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025.
- [38] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36: 68539–68551, 2023.
- [39] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- [40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [41] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [42] Han Shen. On entropy control in llm-rl algorithms. *arXiv preprint arXiv:2509.03493*, 2025.
- [43] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.
- [44] Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>
- [45] RUCAIBox STILL Team. Still-3-1.5b-preview: Enhancing slow thinking abilities of small models through reinforcement learning. 2025. URL https://github.com/RUCAIBox/Slow_Thinking_with_LLMs

- [46] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL <https://github.com/huggingface/trl>
- [47] Hanyin Wang, Zhenbang Wu, Gururaj J. Kolar, Hariprasad Reddy Korsapati, Brian Bartlett, Bryan Hull, and Jimeng Sun. Reinforcement learning for out-of-distribution reasoning in LLMs: An empirical study on diagnosis-related group coding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=0jvnfH0WYV>.
- [48] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [49] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.
- [50] Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025.
- [51] Haofeng Xu, Junwei Su, Yukun Tian, Lansong Diao, Zhengping Qian, and Chuan Wu. Gac: Stabilizing asynchronous rl training for llms via gradient alignment control. *arXiv preprint arXiv:2603.01501*, 2026.
- [52] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- [53] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Bb4VG0WELI>.
- [54] Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. Table-rl: Inference-time scaling for table reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20616–20635, 2025.
- [55] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncal Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=2a36EMSSTp>.
- [56] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun MA, and Junxian He. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=vSMCBUgrQj>.
- [57] Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.
- [58] Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, et al. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*, 2025.
- [59] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.
- [60] Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. <https://github.com/THUDM/slime>, 2025. GitHub repository. Corresponding author: Xin Lv.

Outline of the Appendix:

- Section [A](#): Further Experiment Results
 - Section [A.1](#): Evaluation Results with Standard Deviation
 - Section [A.2](#): Does Prompt Augmentation Increase Token-level Policy Entropy?
 - Section [A.3](#): Increasing KL Regularization Does Not Match PrAg-PO
- Section [B](#): Prompts and Format Reward Details
 - Section [B.1](#): Template Statistics Visualization
 - Section [B.2](#): 13 Templates for Qwen2.5 and Qwen3
 - Section [B.3](#): 35 Templates for DeepSeek-R1
 - Section [B.4](#): Format Reward Best Practices

A Further Experiment Results

A.1 Evaluation Results with Standard Deviation

Table 4: Precise evaluation statistics for Table 1 with standard deviation over the 5 inference rounds.

Method	AIME24	AMC	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
GRPO	23.33 $\pm$ 0.00	49.88 $\pm$ 1.37	75.36 $\pm$ 0.79	26.32 $\pm$ 0.49	38.28 $\pm$ 0.31	42.64 $\pm$ 0.40	48.41 $\pm$ 0.42
DAPO	23.33 $\pm$ 0.00	54.94 $\pm$ 1.37	76.92 $\pm$ 0.54	25.96 $\pm$ 0.56	39.44 $\pm$ 0.32	44.12 $\pm$ 0.37	49.62 $\pm$ 0.33
PrAg-PO (Step 2720)	23.33 $\pm$ 0.00	53.49 $\pm$ 1.08	78.40 $\pm$ 0.45	31.25 $\pm$ 0.37	39.41 $\pm$ 0.66	45.18 $\pm$ 0.19	50.92 $\pm$ 0.30
PrAg-PO (Step 2480)	20.00 $\pm$ 0.00	49.40 $\pm$ 1.20	80.40 $\pm$ 0.32	29.71 $\pm$ 0.48	41.24 $\pm$ 0.59	44.15 $\pm$ 0.32	51.81 $\pm$ 0.28

Table 5: Precise evaluation statistics for Table 2 with standard deviation over the 5 inference rounds..

Method	AIME24	AMC23	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
GRPO	30.00 $\pm$ 0.00	74.00 $\pm$ 1.37	85.48 $\pm$ 0.18	28.24 $\pm$ 0.71	46.49 $\pm$ 0.12	52.84 $\pm$ 0.28	56.47 $\pm$ 0.14
DAPO	33.33 $\pm$ 0.00	77.50 $\pm$ 1.77	84.24 $\pm$ 0.97	26.91 $\pm$ 1.57	50.04 $\pm$ 0.91	54.41 $\pm$ 0.34	57.56 $\pm$ 0.46
PrAg-PO (Step 1100)	42.00 $\pm$ 2.98	78.50 $\pm$ 3.35	87.48 $\pm$ 0.73	29.34 $\pm$ 0.92	53.81 $\pm$ 0.64	58.23 $\pm$ 1.10	60.94 $\pm$ 0.18
PrAg-PO (Step 1160)	33.33 $\pm$ 0.00	77.00 $\pm$ 2.74	87.24 $\pm$ 1.05	32.65 $\pm$ 0.16	56.18 $\pm$ 0.83	57.28 $\pm$ 0.84	62.29 $\pm$ 0.63

Table 6: Precise evaluation statistics for Table 3 with standard deviation over the 5 inference rounds..

Method	AIME24	AMC23	MATH500	Minerva	Olympiad	AVG benchmark	AVG question
GRPO	16.67 $\pm$ 0.00	70.00 $\pm$ 0.00	78.72 $\pm$ 0.87	33.16 $\pm$ 0.55	45.13 $\pm$ 0.37	48.73 $\pm$ 0.13	54.15 $\pm$ 0.29
DAPO	26.67 $\pm$ 0.00	70.00 $\pm$ 0.00	84.88 $\pm$ 0.39	32.79 $\pm$ 0.66	55.05 $\pm$ 0.13	53.88 $\pm$ 0.16	60.73 $\pm$ 0.20
PrAg-PO (Step 1520)	33.33 $\pm$ 0.00	75.00 $\pm$ 0.00	85.12 $\pm$ 0.23	33.60 $\pm$ 0.56	53.87 $\pm$ 0.13	56.18 $\pm$ 0.12	60.69 $\pm$ 0.13
PrAg-PO (Step 720)	23.33 $\pm$ 0.00	76.50 $\pm$ 1.37	87.04 $\pm$ 0.43	33.53 $\pm$ 1.20	55.67 $\pm$ 0.44	55.22 $\pm$ 0.13	61.95 $\pm$ 0.25

A.2 Does Prompt Augmentation Increase Token-level Policy Entropy?

Given that prompt augmentation elicits diverse reasoning paths and output formats, a natural question is whether it also regularizes training by increasing token-level policy entropy. We find that the answer is nuanced: **As shown in Figure 7, Figure 8, and Figure 9, prompt augmentation helps stabilizing policy entropy instead of merely increasing it.**

For a large language model parameterized by θ , the policy π_θ defines a conditional distribution over the next token given the input question q and the previously generated tokens $o_{<t}$. The token-level policy entropy at decoding step t is defined as:

$$\mathcal{H}_t(\pi_\theta) = - \sum_{v \in \mathcal{V}} \pi_\theta(v \mid q, o_{<t}) \log \pi_\theta(v \mid q, o_{<t}),$$

where $\mathcal{V}$ denotes the vocabulary. In practice, token-level policy entropy aggregated over generated response tokens is computed as

$$\mathcal{H}_{\text{agg}}(\pi_\theta) = \frac{1}{\sum_{i=1}^B |o_i|} \sum_{i=1}^B \sum_{t=1}^{|o_i|} \mathcal{H}_{i,t}(\pi_\theta),$$

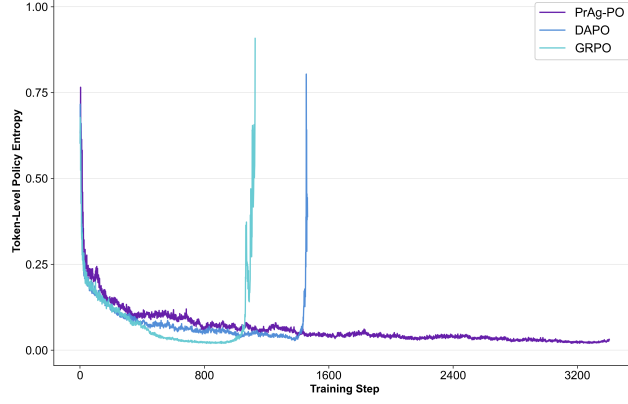


Figure 7: Token-level policy entropy during training of different methods on Qwen2.5-Math-1.5B.

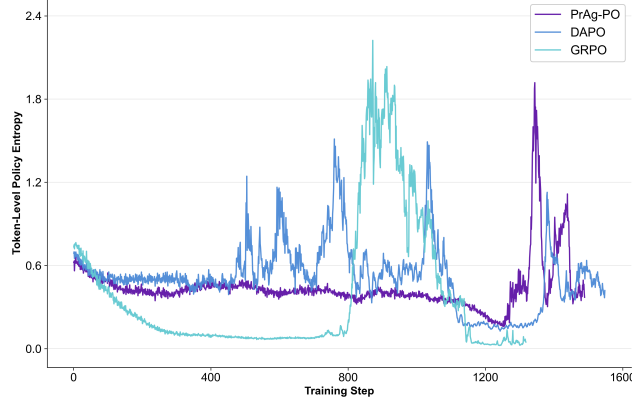


Figure 8: Token-level policy entropy during training of different methods on DeepSeek-R1-Distill-Qwen-1.5B

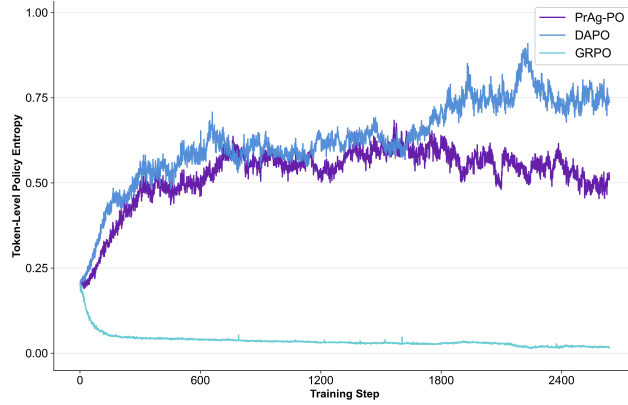


Figure 9: Token-level policy entropy during training of different methods on Qwen3-1.7B

where o_i denotes the i -th generated response in a batch B . Higher entropy corresponds to more stochastic and exploratory generation behavior. However, while token-level entropy measures **local uncertainty**, true diversity is inherently a sequence-level property. In principle, sequence-level entropy provides a more faithful characterization of this diversity, but it is intractable to compute for autoregressive models.

As illustrated in Figure 7, Figure 8, and Figure 9, training collapse is often accompanied by sharp entropy spikes and unstable fluctuations. While the token-level policy entropy of PrAg-PO (purple)

is not consistently higher than that of DAPO, its entropy curve is notably smoother and exhibits a delayed onset of large fluctuations. This suggests that prompt augmentation stabilizes token-level policy entropy during training, rather than simply increasing it, while still promoting diversity through variation in reasoning paths and output formats.

A.3 Increasing KL Regularization Does Not Match PrAg-PO

Strong KL regularization constrains the model within narrower trust regions and has the potential to stabilize training, mitigate entropy collapse, and enable long-horizon optimization. Accordingly, we conduct GRPO training with a substantially larger KL coefficient ($\beta = 0.04$, as originally used in TRL) over an extended training horizon. As shown in Figure 10, although strong KL regularization effectively delays training collapse, it yields significantly lower performance than PrAg-PO. This is because strong KL constraints restrict exploratory behavior and keep the policy overly close to the base model. These results further highlight the effectiveness of PrAg-PO.

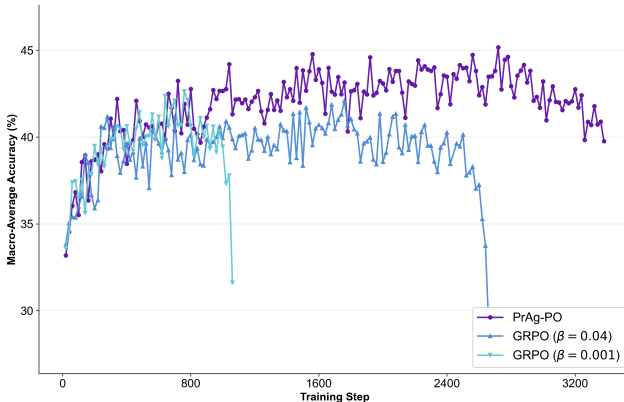


Figure 10: Evaluation accuracy over training steps with varying KL regularization coefficient. Increasing the KL coefficient in GRPO (darker blue) delays collapse but yields substantially lower accuracy.

B Prompts and Format Rewards Details

B.1 Template Statistics Visualization

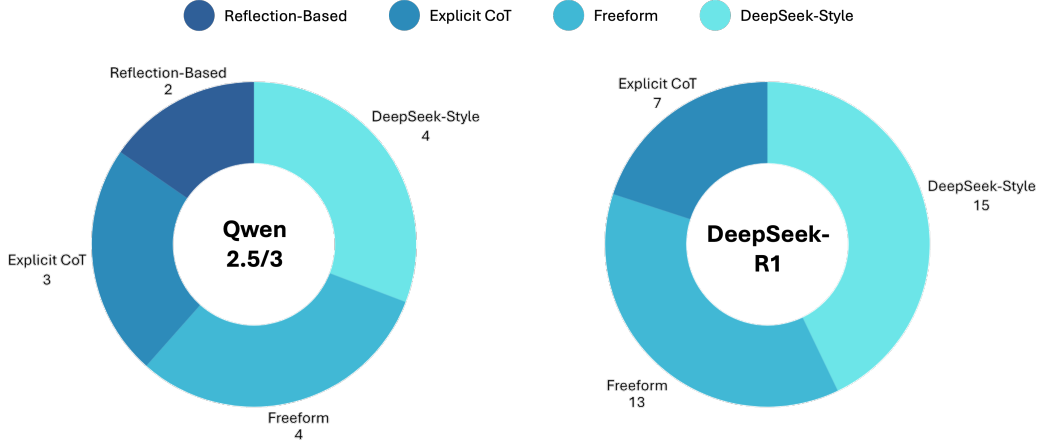


Figure 11: Template category distribution for Qwen2.5-Math-1.5B and Qwen3-1.7B (left), and DeepSeek-R1-Distill-Qwen-1.5B (right). Reflective templates are removed for DeepSeek-R1 because we observe that they slightly degrade training accuracy. We observe that DeepSeek-R1-Distill-Qwen-1.5B already exhibits frequent reflection and self-doubt behavior, and training with reflective templates can make this behavior excessive.

B.2 13 Templates for Qwen2.5 and Qwen3

Table 7: 13 initial prompt templates used to train Qwen2.5-Math-1.5B and Qwen3-1.7B models, their associated categorizations, and format reward functions.

Prompt Template	Category	Format Reward Function
<code><lim_start>system</code> Please reason step by step, and put your final answer within <code>\boxed{}</code> . <code><lim_end></code> <code><lim_start>user</code> {question} <code><lim_end></code> <code><lim_start>assistant</code> Let’s think step by step.	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<code><lim_start>system</code> You are a helpful assistant. <code><lim_end></code> <code><lim_start>user</code> {question} Please reason step by step, and put your final answer within <code>\boxed{}</code> . <code><lim_end></code> <code><lim_start>assistant</code>	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<code><lim_start>system</code> Please reason step by step, and put your final answer within <code>\boxed{}</code> . <code><lim_end></code> <code><lim_start>user</code> {question} <code><lim_end></code> <code><lim_start>assistant</code>	Freeform	<pre>def format_reward(completion): return 1.0</pre>

<lim_start>system You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think> ... </think> <answer> ... </answer> Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>\n") == 1: count += 0.25 if completion.count("\n</think>\n") == 1: count += 0.25 if completion.count("\n<answer>\n") == 1: count += 0.25 if completion.count("\n</answer>") == 1: count += 0.25 return count</pre>
<lim_start>system You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think> ... </think> <answer> ... </answer> Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant </think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("\n</think>\n") == 1: count += 1/3 if completion.count("\n<answer>\n") == 1: count += 1/3 if completion.count("\n</answer>") == 1: count += 1/3 return count</pre>
<lim_start>system A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within <think> </think> and answer is enclosed within <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.25 if completion.count("</think>") == 1: count += 0.25 if completion.count("<answer>") == 1: count += 0.25 if completion.count("</answer>") == 1: count += 0.25 return count</pre>

<lim_start>system A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within <think> </think> and answer is enclosed within <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant <think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1/3 if completion.count("<answer>") == 1: count += 1/3 if completion.count("</answer>") == 1: count += 1/3 return count</pre>
<lim_start>system You are an intelligent assistant who helps with user questions. Provide a rigorous, step-by-step derivation of the solution. The final answer must be clearly indicated within \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<lim_start>system Solve the following math challenge. Explain your approach step-by-step The answer should end with: The final answer is: \boxed{answer} where [answer] is just the final number or expression that solves the problem.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant Let's think step by step	Explicit CoT	<pre>def format_reward(completion): if completion.count("The final answer is:") == 1: return 1.0 else: return 0.0</pre>
<lim_start>system Analyze and solve the math task.<lim_end> <lim_start>user {question} End the answer with: The final answer is: \boxed{answer} where [answer] is just the final number or expression that solves the problem.<lim_end> <lim_start>assistant	Freeform	<pre>def format_reward(completion): if completion.count("The final answer is:") == 1: return 1.0 else: return 0.0</pre>

<lim_start>system Solve the following math problem Show each step of your solution Put the final answer within \boxed{answer} where [answer] is just the final number or expression that solves the prob- lem.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant Let's think step by step	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<lim_start>system You are a helpful assistant that solves math problems. Always write out your reasoning to produce a solution, then check whether the solution is correct, fix it if it is wrong, and finally give the final answer. Respond in exactly the following format: <solution> reasoning and solution </solution> <check> Let's verify step by step ... </check> Put your final answer within \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant	Reflection- Based	<pre>def format_reward(completion): count = 0.0 if completion.count("<solution >\n") == 1: count += 0.25 if completion.count("<\n</ solution>\n") == 1: count += 0.25 if completion.count("<\n<check >\n Let's verify step by step") == 1: count += 0.25 if completion.count("</answer> ") == 1: count += 0.25 return count</pre>
<lim_start>system You are a helpful assistant that solves math problems. Always write out your reasoning to produce a solution, then check whether the solution is correct, fix it if it is wrong, and finally give the final answer. Respond in exactly the following format: <solution> reasoning and solution </solution> <check> Let's verify step by step ... </check> Put your final answer within \boxed{ }.<lim_end> <lim_start>user {question}<lim_end> <lim_start>assistant <solution>	Reflection- Based (Teacher- Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("<\n</ solution>\n") == 1: count += 1/3 if completion.count("<\n<check >\n Let's verify step by step") == 1: count += 1/3 if completion.count("<\n</check >") == 1: count += 1/3 return count</pre>

B.3 35 Templates for DeepSeek-R1

Table 8: 35 prompt templates used to train the DeepSeek-R1-Distill-Qwen-1.5B model, their associated categorizations, and format reward functions. Note that DeepSeek-R1 use a different default template than Qwen, so we first map the initial Qwen templates to DeepSeek style (e.g., ‘assistant’ to ‘<Assistant>’). The first 11 templates come from the initial template set, and the next 23 templates are generated by a proprietary LLM. The last template is the default template of DeepSeek-R1-Distill-Qwen-1.5B. We removed reflective templates for this training because we found that they slightly degraded training accuracy. We observe that DeepSeek-R1-Distill-Qwen-1.5B already exhibits frequent reflection and self-doubt behavior, and training with reflective templates can make this behavior excessive. We add line breaks for presentation clarity; please refer to our codebase for the exact template strings.

Prompt Template	Category	Format Reward Function
<bos>Please reason step by step, and put your final answer within \boxed{ }. <User>{question} <Assistant>Let’s think step by step.	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<bos>You are a helpful assistant. <User>{question} Please reason step by step, and put your final answer within \boxed{ }. <Assistant>	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Please reason step by step, and put your final answer within \boxed{ }. <User>{question} <Assistant>	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think> ... </think> <answer> ... </answer> Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{ }. <User>{question} <Assistant>	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>\n") == 1: count += 0.25 if completion.count("\n</think>\n") == 1: count += 0.25 if completion.count("\n<answer>\n") == 1: count += 0.25 if completion.count("\n</answer>\n") == 1: count += 0.25 return count</pre>
<bos>You are a helpful AI Assistant that provides well-reasoned and detailed responses. You first think about the reasoning process as an internal monologue and then provide the user with the answer. Respond in the following format: <think> ... </think> <answer> ... </answer> Inside the <answer>...</answer>block, the final answer must be enclosed in \boxed{ }. <User>{question} <Assistant><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("\n</think>\n") == 1: count += 1/3 if completion.count("\n<answer>\n") == 1: count += 1/3 if completion.count("\n</answer>\n") == 1: count += 1/3 return count</pre>

<bos>A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within <think></think> and answer is enclosed within <answer></answer> tags, respectively, i.e., <think>reasoning process here </think><answer>answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{ }. <User!>{question} <Assistant!>	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.25 if completion.count("</think>") == 1: count += 0.25 if completion.count("<answer>") == 1: count += 0.25 if completion.count("</answer>") == 1: count += 0.25 return count</pre>
<bos>A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within <think></think> and answer is enclosed within <answer></answer> tags, respectively, i.e., <think>reasoning process here </think><answer>answer here </answer>. Inside the <answer>...</answer> block, the final answer must be enclosed in \boxed{ }. <User!>{question} <Assistant!><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1/3 if completion.count("<answer>") == 1: count += 1/3 if completion.count("</answer>") == 1: count += 1/3 return count</pre>
<bos>You are an intelligent assistant who helps with user questions. Provide a rigorous, step-by-step derivation of the solution. The final answer must be clearly indicated within \boxed{ }. <User!>{question} <Assistant!>	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Solve the following math challenge. Explain your approach step-by-step. The answer should end with: The final answer is: \boxed{answer} where [answer] is just the final number or expression that solves the problem <User!>{question} <Assistant!>Let's think step by step	Explicit CoT	<pre>def format_reward(completion): key = "The final answer is:" format_score = 1.0 if completion.count(key) == 1 else 0.0 return format_score</pre>
<bos>Analyze and solve the math task. <User!>{question} End the answer with: The final answer is: \boxed{answer} where [answer] is just the final number or expression that solves the problem. <Assistant!>	Freeform	<pre>def format_reward(completion): key = "The final answer is:" format_score = 1.0 if completion.count(key) == 1 else 0.0 return format_score</pre>
<bos>Solve the following math problem. Show each step of your solution. Put the final answer within \boxed{ } <User!>{question} <Assistant!>Let's think step by step	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>

<bos>Solve step by step and ensure the final answer is enclosed in \boxed{ }. < User >{question} < Assistant >Let's solve this step by step.	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<bos>Carefully derive the solution step by step. Final answer must be in \boxed{ }. < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Provide a detailed reasoning process. Conclude with \boxed{ }. < User >{question} < Assistant >Start reasoning:	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<bos>Break the problem into steps and solve sequentially. Final answer in \boxed{ }. < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Follow strict reasoning. Do not skip steps. Final answer must be in \boxed{ }. < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Explain clearly and end precisely with \boxed{ }. < User >{question} < Assistant >Proceed carefully:	Explicit CoT	<pre>def format_reward(completion): return 1.0</pre>
<bos>Ensure logical correctness at every step. Final answer strictly inside \boxed{ }. < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Solve the problem and show reasoning. End with: We conclude that the answer is: \boxed{answer} < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): key = "We conclude that the answer is:" return 1.0 if completion.count(key) == 1 else 0.0</pre>
<bos>Work through the math problem step by step. Finish with: Therefore, the final answer is \boxed{answer} < User >{question} < Assistant >Let's think carefully.	Explicit CoT	<pre>def format_reward(completion): key = "Therefore, the final answer is" return 1.0 if completion.count(key) == 1 else 0.0</pre>
<bos>Provide reasoning and ensure the answer ends with: Based on the analysis, the answer is: \boxed{answer} < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): key = "Based on the analysis, the answer is:" return 1.0 if completion.count(key) == 1 else 0.0</pre>
<bos>Explain your reasoning step by step, and conclude with a final statement of the form: From the steps above, the answer is \boxed{answer} < User >{question} < Assistant >	Freeform	<pre>def format_reward(completion): key = "From the steps above, the answer is" return 1.0 if completion.count(key) == 1 else 0.0</pre>

<bos>Provide concise but sufficient reasoning. Final answer must be in \boxed{ }. <User>{question} <Assistant>	(Concise) Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Solve briefly but include key steps. End with \boxed{ }. <User>{question} <Assistant>	(Concise) Freeform	<pre>def format_reward(completion): return 1.0</pre>
<bos>Solve the following problem. Format: <think> ... </think> <answer> \boxed{ } </answer> Ensure correctness and proper formatting. <User>{question} <Assistant><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("<n</think>n") == 1: count += 1/3 if completion.count("<n<answer>n") == 1: count += 1/3 if completion.count("<n</answer>>") == 1: count += 1/3 return count</pre>
<bos>Solve the following problem. Format: <think> ... </think> <answer> \boxed{ } </answer> Ensure correctness and proper formatting. <User>{question} <Assistant>	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>n") == 1: count += 0.25 if completion.count("<n</think>n") == 1: count += 0.25 if completion.count("<n<answer>n") == 1: count += 0.25 if completion.count("<n</answer>>") == 1: count += 0.25 return count</pre>
<bos>Solve the problem carefully and ensure correctness. Use this exact format: <think>...</think><answer>...</answer> The answer must be enclosed in \boxed{ }. All tags must be properly closed. <User>{question} <Assistant><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("<n</think>n") == 1: count += 1/3 if completion.count("<answer>n") == 1: count += 1/3 if completion.count("<n</answer>") == 1: count += 1/3 return count</pre>

<bos>Solve the problem carefully and ensure correctness. Use this exact format: <think>...</think><answer>...</answer> The answer must be enclosed in \boxed{ }. All tags must be properly closed. <User>{question} <Assistant>	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.25 if completion.count("</think>") == 1: count += 0.25 if completion.count("<answer>") == 1: count += 0.25 if completion.count("</answer>") == 1: count += 0.25 return count</pre>
<bos>You are given a problem to solve. Use this structure exactly: <think> ... </think> <answer> ... </answer> Final answer must be enclosed in \boxed{ }. Do not skip steps. <User>{question} <Assistant><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1/3 if completion.count("<answer>") == 1: count += 1/3 if completion.count("</answer>") == 1: count += 1/3 return count</pre>
<bos>You are given a problem to solve. Use this structure exactly: <think> ... </think> <answer> ... </answer> Final answer must be enclosed in \boxed{ }. Do not skip steps. <User>{question} <Assistant>	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.25 if completion.count("</think>") == 1: count += 0.25 if completion.count("<answer>") == 1: count += 0.25 if completion.count("</answer>") == 1: count += 0.25 return count</pre>
<bos>Your task is to solve the given problem correctly. All reasoning must be inside <think></think>. The final answer must be enclosed in \boxed{ }. Ensure correctness before answering. <User>{question} <Assistant><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1 return count</pre>

<bos>Your task is to solve the given problem correctly. All reasoning must be inside <think></think>. The final answer must be enclosed in \boxed{ }. Ensure correctness before answering. < User >{question} < Assistant >	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.5 if completion.count("</think>") == 1: count += 0.5 return count</pre>
<bos>Solve the problem step by step. Organize your reasoning clearly inside <think></think>. The final answer must be in \boxed{ }. < User >{question} < Assistant ><think>	DeepSeek-Style (Teacher-Forced)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1 return count</pre>
<bos>Solve the problem step by step. Organize your reasoning clearly inside <think></think>. The final answer must be in \boxed{ }. < User >{question} < Assistant >	DeepSeek-Style	<pre>def format_reward(completion): count = 0.0 if completion.count("<think>") == 1: count += 0.5 if completion.count("</think>") == 1: count += 0.5 return count</pre>
<bos>Please reason step by step, and put your final answer within \boxed{ }. < User >{question} < Assistant ><think>	DeepSeek-style (Default Template)	<pre>def format_reward(completion): count = 0.0 if completion.count("</think>") == 1: count += 1 return count</pre>

B.4 Format Reward Best Practices

Empirically, we find that simple additive format rewards often achieve the best performance. In contrast, more complex conditional logic or regular-expression-based rewards can degrade reasoning accuracy. In particular, rules that overly restrict the model’s behavior may accelerate training collapse and lead to worse outcomes. For example, the format reward in Listing 49 limits the number of characters generated after the model’s final boxed answer to fewer than 70, with the goal of preventing the model from continuing to output gibberish after completing its answer. However, we find that this reward structure can induce early collapse and substantially degrade performance.

```
def find_matching_brace(s, start):
    depth = 1 # assumes s[start] == "{"
    for i in range(start + 1, len(s)):
        c = s[i]
        if c == "{":
            depth += 1
        elif c == "}":
            depth -= 1
            if depth == 0:
                return i
    return -1

def boxed_format_and_length_penalty(completion):
    key = r"\boxed"
    idx = completion.rfind(key)
    if idx == -1:
        return 0.0

    # find opening brace after \boxed
    brace_start = completion.find("{", idx)
    if brace_start == -1:
        return 0.0

    # find matching closing brace
    brace_end = find_matching_brace(completion, brace_start)
    if brace_end == -1:
        return 0.0

    # count chars after boxed expression
    tail_len = len(completion) - (brace_end + 1)

    if tail_len > 70:
        return 0.0

    return 1.0

# should be very limited words after last boxed
def abel_format_reward(completion):
    return boxed_format_and_length_penalty(completion)
```

Listing 49: `abel_format_reward` is an example of an overly complex and restrictive format reward function, which requires the model to generate fewer than 70 characters after its final answer.

Ditch the Denoiser: Emergence of Noise Robustness in Self-Supervised Learning from Data Curriculum

Wenquan Lu¹

Jiaqi Zhang¹

Hugues Van Assel²

Randall Balestriero¹

¹Brown University ²Genentech
{wenquan_lu, jiaqi_zhang6, randall_balestriero}@brown.edu
van_assel.hugues@gene.com

Abstract

Self-Supervised Learning (SSL) has become a powerful solution to extract rich representations from unlabeled data. Yet, SSL research is mostly focused on clean, curated and high-quality datasets. As a result, applying SSL on noisy data remains a challenge, despite being crucial to applications such as astrophysics, medical imaging, geophysics or finance. In this work, we present a fully self-supervised framework that enables noise-robust representation learning without requiring a denoiser at inference or downstream fine-tuning. Our method first trains an SSL denoiser on noisy data, then uses it to construct a denoised-to-noisy data curriculum (i.e., training first on denoised, then noisy samples) for pretraining a SSL backbone (e.g., DINOv2), combined with a teacher-guided regularization that anchors noisy embeddings to their denoised counterparts. This process encourages the model to internalize noise robustness. Notably, the denoiser can be discarded after pretraining, simplifying deployment. On ImageNet-1k with ViT-B under extreme Gaussian noise ($\sigma = 255$, SNR = 0.72 dB), our method improves linear probing accuracy by 4.8% over DINOv2, demonstrating that denoiser-free robustness can emerge from noise-aware pretraining. The code is available at https://github.com/wenquanlu/noisy_dinov2

1 Introduction

Self-supervised methods like DINOv2 [31, 7, 16, 9] have demonstrated remarkable success by leveraging unlabeled data to learn visual representations able to solve many tasks in zero or few shot manners [3]. However, the performances of those methods heavily rely on the availability of clean and high-quality datasets. As a result, some prior works have focused on building data curation pipelines for self-supervised learning (SSL) [37, 11]. But data curation disregards samples and introduces additional design questions and thus begs the following question:

Can we enable off-the-shelf SSL models like DINOv2 to learn from highly corrupted data?

In real-world scenarios, datasets often contain noise that can severely degrade the performance of learned representations, and clean references required for supervised denoising are typically unavailable due to sensor limitations, privacy constraints, or acquisition costs.

This is especially common in medical imaging [28, 43, 27], astrophysics [30], geological rock imaging [6], remote sensing [17] and finance [29]. For example, the earth radar images taken by ESA Sentinel-1 [14] comes with speckle noise, and the clean data for supervised denoising is difficult to acquire since it requires better sensor and atmospheric conditions. As another example, Sloan Digital Sky Survey (SDSS) imaging in the COSMOS region contains significant sky noise [30] which can obscure faint celestial objects. Despite advancements in SSL, methods like DINOv2, as shown in Figure 5, still yield significant reduction in performance when trained directly on corrupted datasets.

To address the challenges outlined above, a trivial baseline is a denoiser-preprocessed pipeline, which trains a self-supervised denoiser on the noisy data first, then trains the SSL models on the denoised

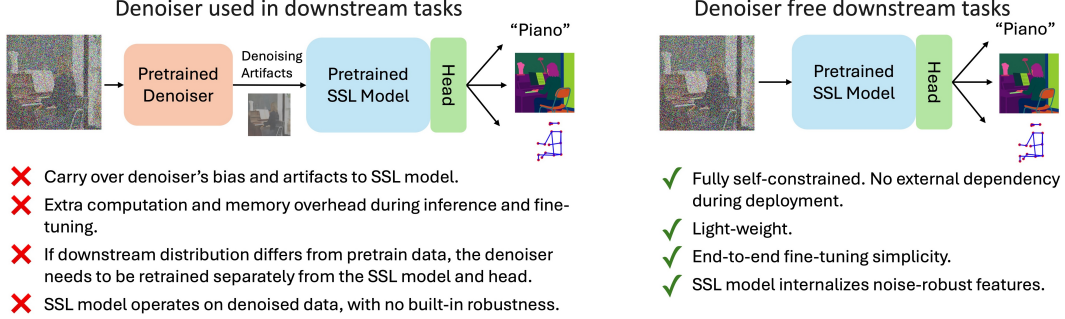


Figure 1: Comparison of downstream pipelines with (left) and without (right) denoisers. The denoiser-free pipeline shows numerous advantages in efficiency, simplicity and robustness.

images. In downstream tasks, the SSL denoiser always serves as a preprocessor because the SSL model is only trained on denoised data thus cannot handle noisy inputs. While effective, this pipeline adds substantial inference latency, fine-tuning overhead, and deployment complexity (Figure 1, left). Instead, we explore training strategies that enable an off-the-shelf SSL backbone (e.g., DINOv2) to ingest noisy inputs directly, thereby ditching the denoiser during downstream use (Figure 1, right). To the best of our knowledge, this problem has little-to-no prior work in the existing literature. We take a first step by corrupting ImageNet [12] with synthetic noise that mirrors real-world degradations, providing a controlled yet challenging testbed for systematic evaluation and benchmarking.

In this paper, we propose a novel, fully self-supervised approach to learn noise-robust representations. Our approach begins similarly as the denoising pipeline baseline: train a SSL denoiser to create a denoised version of the dataset. We then employ a curriculum [5] to train DINOv2 on this denoised dataset for robust feature learning and subsequently restart training on the original noisy dataset to adapt the model to real-world noise. At inference time or downstream task finetuning, the denoiser can be discarded as the SSL model has learned to extract useful representations directly from noisy inputs—without relying on external denoising. In addition, We further introduce a regularization loss using a denoised teacher to guide the student during noisy training, encouraging alignment between noisy and denoised embeddings and improving robustness under extreme noise. Extensive experiments show that our method significantly improves classification and instance recognition performance over DINOv2 trained directly on noisy data. Remarkably, our framework achieves results comparable to, and sometimes exceeding, those obtained by the denoising pipeline baseline. In summary, we make the following contributions:

1. We propose a thorough sensitivity analysis of the latest SOTA SSL method (DINOv2) to noise being present in its pretraining data, a previously unexplored aspect in the literature. We identify that the method is sensitive to its pretraining data quality, especially in low-resource regimes (e.g., short training schedules or smaller datasets).
2. We find that, under mild noise levels, increasing training duration and data size can substantially mitigate performance degradation, but never fully recover the clean-data baseline. This offers actionable guidance for practitioners to deal with noisy data during pretraining.
3. We introduce a new training paradigm for enabling models like DINOv2 to learn robust representations directly from noisy pretraining data. By combining noise curriculum learning with a novel denoised regularization loss, our method achieves fully denoiser-free downstream fine-tuning and inference while maintaining strong performance across tasks. As shown in Figure 2, our methods, DINOv2 w/ NC and DINOv2 w/ NCT improve significantly over the DINOv2 baseline, and closely matches the denoiser-preprocessed pipeline N2N + DINOv2.

2 Related Work

Self-Supervised Learning for Images. A large body of work has focused on learning visual representations using self-supervised methods [15, 46, 33, 9, 10]. Among them, joint-embedding methods have emerged as state-of-the-art, training models to produce similar embeddings for different views of the same image. These methods rely on architectural and loss regularizations to prevent representation collapse, such as enforcing cross-correlation constraints [44] or using momentum

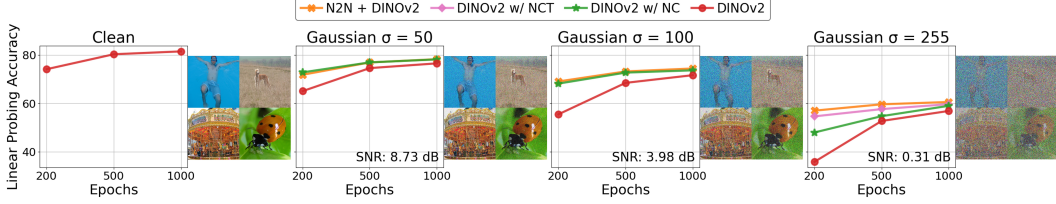


Figure 2: Linear probing accuracies and noise examples on ImageNet-100 dataset. The denoiser-preprocessed baseline N2N + DINOv2 and our denoiser-free methods DINOv2 w/ NC and NCT significantly improve over the DINOv2 baseline. Note that DINOv2 w/ NC and N2N + DINOv2 **overlap** at moderate noise levels, and DINOv2 w/ NCT is only evaluated at Gaussian $\sigma = 255$.

encoders [16, 7, 47, 11]. DINOv2 [31], used in this work, is a joint-embedding method that achieves state-of-the-art performance on ImageNet classification and many evaluation benchmarks.

Self-Supervised Image Denoising learns to denoise noisy images without access to noisy-clean pairs. Noise2Noise [26] uses aligned noisy-noisy pairs derived from the same clean image to train a U-Net [35] model. However, obtaining such pairs is not always practical. Neighbor2Neighbor [20], as used in this paper, mitigates such issue by sampling two subimages from a noisy image which serve as a noisy-noisy pair [32]. It achieves competitive performance on a range of denoising benchmarks [45]. Another line of work uses blind-spot networks (BSN) [24, 8] to learn denoise without clean-image supervision. BSN uses the same noisy image as its inputs to supervise its outputs. It either applies masks in inputs [21, 4, 40] or masks in network structure [22, 42] to prevent model collapse.

Self-Supervised Representation Learning for Noisy Data. There is very limited work on learning noise-robust features for image data. Most of the past works concern noisy time-series data, such as speech [25, 2, 39, 49] and EEG data [48]. [48] demonstrates applying time series SSL methods on noisy datasets yields poor performance, and proposes denoiser-driven contrastive learning by using a conventional denoiser to create noised-conditioned positive and negative pairs. This approach of leveraging a denoiser to learn noise-robust representation shares similar motivations as our method.

3 Noise Robust Self-Supervised Learning

In this section, we introduce our self-supervised method for robust representation learning on noisy image data. We start with a two-stage baseline, then moving towards fully self-supervised, denoiser-free noise robustness through curriculum learning and regularization.

3.1 Preliminary: Curriculum Learning Background and Denoiser-Preprocessed Baseline

Curriculum learning refers to a training scheme that trains a machine learning model from easier to harder data that imitates the learning order in human curricula. This often leads to faster convergence and better performance. The seminal work [5] formally defines a curriculum as a sequence of training distributions $\langle Q_1, \dots, Q_t, \dots, Q_T \rangle$. Each distribution Q_t is a reweighting of the target training distribution $P(z)$: $Q_t(z) \propto W_t(z)P(z) \quad \forall z : \int Q_t(z)dz = 1$ such that following conditions are satisfied: (1) the entropy of distributions increase over time, i.e., $H(Q_t) < H(Q_{t+1})$. This implies that the model sees more diverse or complex examples as training progresses. A rising entropy indicates reduced certainty and increased difficulty. (2) Unnormalized weighting for any example increases over time, i.e., $W_t(z) \leq W_{t+1}(z) \quad \forall z$. (3) Final distribution equals the target, i.e., $Q_T(z) = P(z)$. In practice, the first condition is most widely preserved, while the latter two are often relaxed to allow more flexible curricula [38]. In this work, we adopt this perspective and design our training schedule to follow an entropy-based curriculum from easy to hard examples.

N2N + DINOv2: Joint Self-Supervised Denoising and Representation Learning. As a starting point, we consider a denoiser-preprocessed baseline, an upper bound that combines a self-supervised denoiser with a joint-embedding SSL model. Given a noisy dataset X , we first train a denoiser f_θ on X and generate a denoised set $X_{\text{denoised}} = f_\theta(X)$. This is used to train a self-supervised learner g_θ , such as DINOv2. While effective, this pipeline requires an explicit denoiser in all downstream tasks. We use Neighbor2Neighbor [20] as the denoiser for its robustness across noise levels, but emphasize that the pipeline is **denoiser-agnostic**, a key benefit also shared by our later proposed methods. In practice, the denoiser should be selected based on the domain-specific noise characteristics.

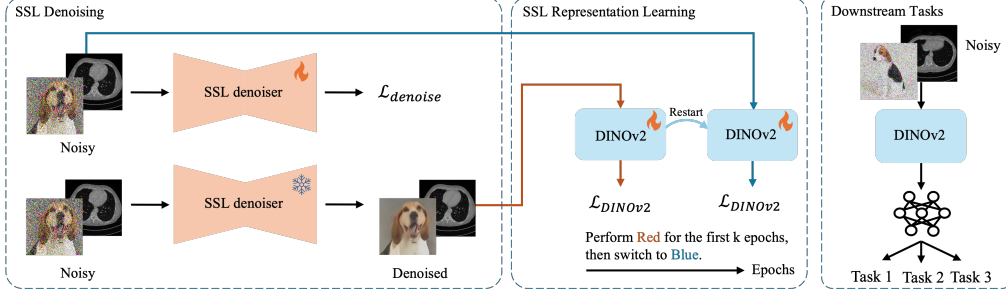


Figure 3: Overview of DINOv2 w/ NC which comprises SSL denoising, SSL representation learning and downstream tasks. A SSL denoiser is trained to denoise the data [28]. Then, DINOv2 is trained on the denoised data for early epochs followed by restarting training on noisy data. Lastly, after some fine-tuning, downstream tasks are performed directly on noisy inputs with an added prediction head.

3.2 DINOv2 w/ NC (Noise Curriculum): Curriculum Learning for Self-Supervised Representation Learning

Ideally, instead of relying on a separate denoiser, the representation model should develop robustness to noise on its own, enabling an **end-to-end** training setup in downstream tasks. Figure 3 outlines our curriculum learning approach to achieve this. Following the definition in Section 3.1, we design a curriculum $\langle Q_1, Q_2 \rangle$ that satisfies the key requirement $H(Q_1) < H(Q_2)$. Here, Q_1 corresponds to the distribution over *denoised* images X_{denoised} , which lies in a more structured, lower-entropy subspace. Q_2 corresponds to the distribution over the original *noisy* images X , which has higher entropy due to the injected stochastic corruption that expands the support of the data distribution.

Such a curriculum is further motivated by the following toy example. We consider a simple MLP with 1 hidden layer trained on MNIST [23] using ReLU activation, and optimize the following self-supervised joint embedding loss Equation (1) that applies a soft identity constraint on empirical covariance matrix, which is a canonical formulation that aligns embeddings of different augmented views while preventing representational collapse:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\tau_1, \tau_2 \sim \mathcal{T}} \|m_{\theta}(\tau_1(x_i)) - m_{\theta}(\tau_2(x_i))\|_2^2 + \lambda \left\| \hat{\Sigma}_{\theta} - I_k \right\|_F^2 \quad (1)$$

where $\hat{\Sigma}_{\theta} = \frac{1}{n} \sum_{i=1}^n (z_i - \bar{z})(z_i - \bar{z})^T$, $z_i = m_{\theta}(\tau(x_i))$, m is the MLP and τ is the augmentation. Under Gaussian noise ($\sigma = 0.4 \cdot 255$), we observed that: Training and linear probing on the clean train set (50 epochs each) yields $91.21 \pm 0.36\%$ (1-sigma) accuracy on the clean test set. Training and linear probing on the noisy train set (50 epochs each) yields $64.55 \pm 3.47\%$ on the noisy test set. Surprisingly, curriculum learning (training 30 epochs on clean, then 20 on noisy), followed by 50 epochs of linear probing on noisy, recovers performance to $83.05 \pm 0.45\%$ on the noisy test set. This shows that the model can adapt to the noisy data without forgetting what it learned from the clean phase. The significant recovery in performance (see Appendix Figure 6), even with a simple MLP and vector data, suggests that curriculum learning encourages the model to internalize noise-robust features. We now build on this intuition to scale our method to complex vision models like DINOv2:

Denoised Pretraining: We start with an initialized self-supervised model g_{θ} (DINOv2) for representation learning. We train g_{θ} on the denoised set X_{denoised} for k epochs to establish a stable initialization that captures the underlying structure of the data while reducing the impact of noise.

Noise Curriculum Transition: Using the weights learned in the first k epochs, we restart training on the original noisy set X until convergence. Here ‘restart’ means re-initializing all training dynamics such as learning rate and weight decay scheduling. This enables the model to better adapt to the noisy dataset, learning noise-robust representations from the stable features learned on the denoised data.

Denoyer-free Downstream Tasks and Inference: For downstream tasks involving noisy images, such as classification, the representations learned by g_{θ} can be directly utilized with a prediction head h_{θ} for task-specific learning. Further fine-tuning can be performed directly on the noisy data z without requiring a denoising module, since the training strategy internalizes noise-robust features into the learned representations. This enables the entire model, both backbone g_{θ} and head h_{θ} , to be optimized end-to-end on the downstream objective and deployed without any preprocessing at inference time as shown in Equation (2):

$$\text{Finetuning: } \mathcal{L} = \mathbb{E}_{(z,y)} [L(h_\theta(g_\theta(z)), y)] \quad \text{Inference: } \hat{y} = h_\theta(g_\theta(z)) \quad (2)$$

This highlights a key advantage of the proposed **DINOv2 w/ NC** as it enables fully denoiser-free downstream fine-tuning and inference. In contrast, the two-stage baseline **N2N + DINOv2** requires a separate self-supervised denoiser trained and applied not only during pretraining but also downstream task fine-tuning and inference. This external dependency adds computational overhead and pipeline complexity, risks transferring the denoiser’s bias to the SSL model, and deviates from our goal of learning noise-robust representations directly from corrupted data.

3.3 DINOv2 w/ NCT (Noise Curriculum Teacher): Anchored Curriculum Learning for Improved Self-Supervised Representation Learning

Under high or extreme noise levels, a good initialization from denoised training may not be sufficient, because strong noise may destabilize and disturb the prior learned representations during the noisy training. Thus, we propose to utilize the embeddings of the denoised images as anchors to regularize the training on noisy data. Let the teacher backbone of DINOv2 be T , and the student backbone be S . The original DINOv2 loss can be expressed as

$$L_{\text{dino}v2} = L_{\text{dino}\&\text{ibot}}(T(\tau_t(x)), S(\tau_s(x))) + L_{\text{koleo}} \quad (3)$$

Where L_{koleo} is the Koleo Regularization, τ_t is the augmentation for teacher input, and τ_s is the augmentation for student input. $L_{\text{dino}\&\text{ibot}}$ is the combined DINO and iBOT loss that minimizes the cross-entropy of image-level and patch-level output scores between teacher $\{p_t^{\text{img}}, p_t^{\text{patch}}\}$ and student $\{p_s^{\text{img}}, p_s^{\text{patch}}\}$. Here the backbones T and S include ‘heads’ that output vector scores.

$$T(\tau_t(x)) = \{p_t^{\text{img}}, p_t^{\text{patch}}\} \quad S(\tau_s(x)) = \{p_s^{\text{img}}, p_s^{\text{patch}}\} \quad (4)$$

$$L_{\text{dino}\&\text{ibot}} = - \sum p_t^{\text{img}} \log p_s^{\text{img}} - \sum p_t^{\text{patch}} \log p_s^{\text{patch}} \quad (5)$$

Before restarting the noisy training, we first extract the weights of the teacher backbone T_{dn} that was trained on denoised data and freeze it. As shown in Figure 4, the architecture of the model now contains three components: teacher T , student S and frozen denoised teacher T_{dn} . When restarting the training, given a noisy image x and its denoised version $x_{\text{dn}} = f_\theta(x)$, we apply exactly identical augmentations (i.e., same crops, blur, flip...) to x and x_{dn} , so their embeddings are aligned. In addition to the original DINOv2 loss, we encourage the output scores of the student to be close to that of the denoised teacher. This prevents the training from deviating significantly from its cleaner initialization. So the denoised-regularized loss is given as:

$$L = L_{\text{dino}v2} + \lambda L_{\text{dino}\&\text{ibot}}(T_{\text{dn}}(\tau_t(x_{\text{dn}})), S(\tau_s(x))) \quad (6)$$

Where λ is a parameter that controls the strength of the regularization. It is worth noting that the weights of T and T_{dn} are identical at the start of the training. Also, the technique presented is not restricted to only denoised settings. When clean images are available, such method can be used to improve the overall noise-robustness of DINOv2 by simply replacing ‘denoised’ components with ‘clean’ components.

It is important to emphasize that this regularization technique is specifically designed to complement our curriculum training pipeline. For the regularization to be effective, the frozen teacher T_{dn} and the trainable teacher T must have alignment in their output embeddings at the beginning of the restart training. This is accomplished by assigning them identical weights. Using a different frozen teacher, even if trained on the same denoised dataset, would not provide meaningful regularization due to misalignments, as shown in Appendix Figure 9. Like NC, NCT is also denoiser-free at downstream fine-tuning and inference.

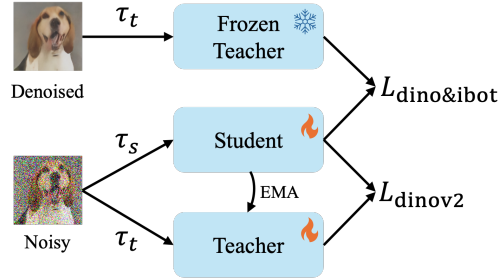


Figure 4: Structure of the denoised regularization loss. In addition to the original DINOv2 loss $L_{\text{dino}v2}$, a regularization loss term $L_{\text{dino}\&\text{ibot}}$ is introduced by comparing the output scores between the student and the frozen teacher. The frozen teacher takes in denoised inputs that undergo identical augmentations τ_t as the noisy inputs of the trainable teacher.

4 Experiments

4.1 Practical and Challenging Experimental Settings

Datasets. We conduct experiments on both ImageNet-100 and ImageNet-1k to evaluate the robustness of SSL methods under noise. ImageNet-100 comprises 100 classes as defined in Mini-ImageNet [36]. The training set includes the first 500 images per class, totaling 50k training and 5k validation images. For large-scale evaluation, we experiment on ImageNet-1k [12] using the full training and validation splits. For instance recognition evaluation, we use the Oxford and Paris dataset [34].

Noise Addition. We follow ImageNet-C [19] to introduce three types of noise: Gaussian, Shot (Poisson), and Speckle noise, as these are among the most commonly encountered noise types in natural imaging. For the ImageNet-100 experiments, we use Gaussian noise ($\sigma = 50, 100, 255$; SNR: 8.73 dB, 3.98 dB, 0.31 dB), Shot noise ($\lambda = 10, 3, 1$; SNR: 8.49 dB, 4.11 dB, 0.52 dB), and Speckle noise ($\sigma = 102, 178.5, 255$; SNR: 9.15 dB, 5.58 dB, 3.98 dB). For the ImageNet-1k experiments, we tested Gaussian noise at $\sigma = 100$ and $\sigma = 255$, with corresponding SNRs of 4.36 dB and 0.72 dB respectively. These SNRs are common values reported in real noisy medical imaging [41]. The example visualizations for each noise level are provided in Section D.2. The noises are added to raw images before preprocessing (e.g., resizing) to better reflect practical scenarios such as device or environmental noise. Consequently, while the initial noise level is fixed, the preprocessing steps introduce variations in actual noise levels that challenge the model’s ability to adapt effectively. Notably, we generate the noise dataset once prior to training and do not apply noise on-the-fly during data loading, which closely simulate real-world scenarios where one starts with a fixed noisy dataset.

Implementation Details. In ImageNet-100 experiments, we use ViT-S/16 as the base architecture for DINOv2, and unless otherwise mentioned, train it for 200 epochs with a batch size of 40 using the AdamW optimizer. The base learning rate is 7.9×10^{-4} (scaled by the square root of batch size). The denoiser is trained for 100 epochs with a batch size of 4 using Adam. The linear probe uses a batch size of 128 and is trained for 12.5k steps. In ImageNet-1k, we adopt ViT-B/16 as the backbone and train DINOv2 for 100 epochs with a batch size of 512 and drop path rate of 0.1. The learning rate is 2.8×10^{-3} . The denoiser is trained on a 100k subset with a batch size of 8 for 50 epochs (for Gaussian $\sigma = 255$) and 90 epochs (for Gaussian $\sigma = 100$). The linear probe uses batch size 1024 for 25k steps. Each training is done on a single RTX 4090 (~8h) or 4×L40S GPUs (~65h). The strength parameters are determined via light parameter sweeps with the details provided in Section A.2.

4.2 Linear Probing Reveals Superior Noise Adaptation

We begin by evaluating how different methods adapt to various levels and types of noise using a 200-epoch training schedule on ImageNet-100. We train a linear classifier over a frozen DINOv2 backbone on the training set, and report classification performance over the validation set. We compare the classification performance of the DINOv2 backbone over four training strategies. (1) **N2N + DINOv2**: train the backbone on denoised images for 200 epochs and evaluate the model on denoised images (2) **DINOv2 w/ NC**: train the backbone on denoised images for 140 epochs, then restart training on noisy images for 60 epochs without regularization. Finally, evaluate the model on noisy images. The rationale for such partitions of epochs is provided in Appendix B.3. (3) **DINOv2 w/ NCT**: same as the DINOv2 w/ NC but with the regularization loss applied in the 60 epochs of noisy training. We only evaluate the NCT

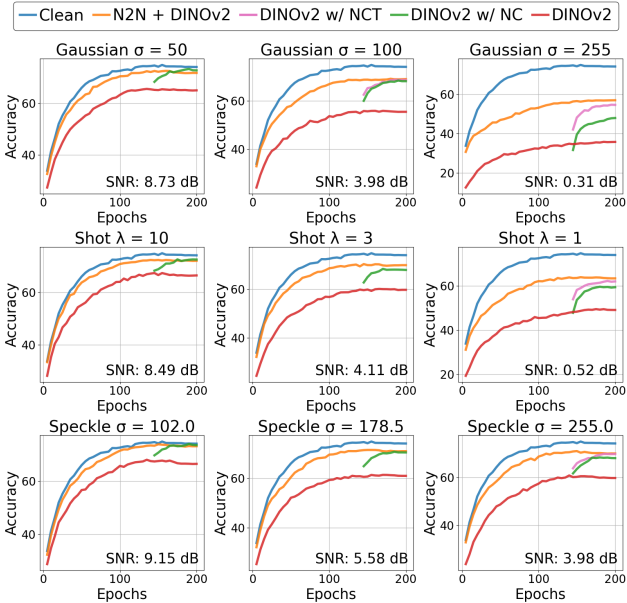


Figure 5: Linear probing classification accuracies on DINOv2 backbone weights saved every 5 epochs during training on ImageNet-100 under various noise levels.

Table 1: Performance comparison of different methods when scaling training duration and data size.

Noise Type	Method	200 Ep.	500 Ep.	1000 Ep.
Clean	DINOv2	74.1	80.3	81.4
Gaussian-50 SNR: 8.73 dB	N2N + DINOv2	71.8	76.9	78.2
	DINOv2 w/ NC	72.8	76.9	78.1
	DINOv2	65.0	74.5	76.5
Gaussian-100 SNR: 3.98 dB	N2N + DINOv2	69.0	73.2	74.4
	DINOv2 w/ NC	68.1	72.7	73.6
	DINOv2	55.4	68.4	71.6
Gaussian-255 SNR: 0.31 dB	N2N + DINOv2	57.0 [†]	59.6 [‡]	60.5
	DINOv2 w/ NCT	54.6	57.6 [†]	59.6 [‡]
	DINOv2 w/ NC	47.9	54.7	59.0
	DINOv2	35.8	52.7	56.8

(a) Evolution of linear probing accuracies on ImageNet-100 under extended training durations. **Relative ranking among methods remain consistent across epochs.** See visualization in Figure 2.

method on the four most noisy scenarios (lowest SNRs), i.e., Gaussian noise ($\sigma=255$, $\sigma=100$), Shot noise ($\lambda=1$), and Speckle noise ($\sigma=255$). (4) **DINOv2**: train the backbone on noisy images for 200 epochs, and evaluate the model on noisy images. Additionally, we train and evaluate the model on original clean images, referred to as **Clean**. Figure 5 illustrates the classification accuracy curves.

As shown in Figure 5, our proposed method (green) significantly outperforms the backbone trained solely on noisy images (red) across all noise types and noise levels. At moderate noise levels, i.e., the first two columns, the accuracy of our method closely matches and even exceeds the backbone that is trained and evaluated on explicitly denoised data (shown in orange lines). Only at extreme noise levels, i.e., the last column, the gap between **DINOv2 w/ NC** and **N2N + DINOv2** widens, primarily due to the significant loss of the original image signal. For instance, as shown in Section D.2 with Gaussian noise at $\sigma = 255$, many images become virtually unrecognizable to the naked eye. Hence, regularization is introduced to guide the model in these extremely noisy scenarios. As illustrated in Figure 5, regularization **DINOv2 w/ NCT** improves the model’s performance at all four scenarios. The improvement amplifies as the gap between **N2N + DINOv2** and **DINOv2 w/ NC** widens, with a substantial improvement of 6.7% over **NC** achieved at Gaussian noise with $\sigma = 255$. However, when **N2N + DINOv2** is on par with **DINOv2 w/ NC**, the regularization provides diminishing benefit, as demonstrated with Gaussian noise at $\sigma = 100$. This is expected since the guidance becomes redundant when its representation quality is comparable to or lower than that of the original model.

4.3 Scaling Improves Performance Across Datasets

Next, we scale both training and data size to evaluate performance in more resource-rich settings.

Longer Training Duration. We further improve the performance of our methods by extending the training on ImageNet-100 to longer epochs. For 500-epoch configuration, we restart the training from the 200-epoch denoised checkpoint (i.e., **N2N + DINOv2**) as described in Section 4.2, for the 1000-epoch configuration, we restart the training from the 500-epoch denoised checkpoint. As shown in Table 1a and Figure 2, accuracies increase as epoch increases and the relative ranking among methods remains consistent across training durations: **N2N + DINOv2** achieves the highest accuracy, followed closely by **DINOv2 w/ NC or NCT**, while the baseline **DINOv2** remains the lowest. This demonstrates the scalability of the proposed methods. Interestingly, we observe that **DINOv2 w/ NCT** consistently converges to or outperforms its anchor’s accuracy, as indicated by [†] and [‡] in Table 1a demonstrating that the regularization objective successfully guides the model toward robust representations. Notably, the performance gap between **DINOv2** and the other methods narrows as training progresses, suggesting that **DINOv2** is sensitive to noise and benefits significantly from longer training. In contrast, our noise curriculum and regularization approach greatly accelerates convergence, achieving similar or better performance in roughly half the training time.

Larger Dataset. In our controlled ImageNet-1k experiment, we constrain the **DINOv2** backbone to undergo a total of 100 training epochs to enable fair comparison across methods. **DINOv2 w/ NC**: train the backbone on denoised images for 30 epochs followed by 70 epochs of training on noisy images. **DINOv2 w/ NCT**: While we find that restarting at 30 epochs yields the best performance for **DINOv2 w/ NC**, the denoised teacher at this early stage lacks sufficiently strong representations to serve as an effective regularizer. To address this, we continue training the denoised teacher to 100 epochs, then freeze it for regularizing the **DINOv2** backbone for 70 epochs of noisy training. Since

Method	Clean	Gaussian-100 SNR: 4.36 dB	Gaussian-255 SNR: 0.72 dB
N2N + DINOv2	-	73.1	57.2
DINOv2 w/ NCT	-	72.1	55.8
DINOv2 w/ NC	-	70.9	53.5
DINOv2	79.0	70.7	51.0
SimCLR	60.1	48.0	34.2

(b) Linear probing classification accuracy on ImageNet-1k under Gaussian noise using a 100-epoch training schedule. **DINOv2 w/ NCT** and **DINOv2 w/ NC** show substantial improvements over the **DINOv2** baseline, particularly at high noise levels ($\sigma = 255$). SimCLR is included here as a reference baseline.

Table 2: Comparison of linear probing accuracies on clean and noisy validation sets for models trained on noisy ImageNet-1k and ImageNet-100 datasets. **Surprisingly, our DINOv2 w/ NC or NCT method outperforms the two-stage N2N + DINOv2 on clean validation sets**, demonstrating more accurate and generalizable representations.

Dataset	Training Noise Method	Clean	Noisy
ImageNet-1k (100 epochs)	Gaussian-100 SNR: 4.36 dB	N2N + DINOv2	75.8 73.1
		DINOv2 w/ NCT	75.2 72.1
		DINOv2 w/ NC	74.3 70.9
		DINOv2	74.1 70.7
	Gaussian-255 SNR: 0.72 dB	N2N + DINOv2	64.1 57.2
		DINOv2 w/ NCT	65.6 55.8
		DINOv2 w/ NC	63.7 53.5
		DINOv2	61.6 51.0
	Gaussian-50 SNR: 8.73 dB	N2N + DINOv2	78.5 78.2
		DINOv2 w/ NC	79.4 78.1
		DINOv2	78.8 76.5
ImageNet-100 (1000 epochs)	Gaussian-100 SNR: 3.98 dB	N2N + DINOv2	76.0 74.4
		DINOv2 w/ NC	76.7 73.6
		DINOv2	74.6 71.6
		N2N + DINOv2	56.7 60.5
	Gaussian-255 SNR: 0.31 dB	DINOv2 w/ NCT	59.4 59.6
		DINOv2 w/ NC	57.9 59.0
		DINOv2	58.8 56.8

Table 3: Linear evaluation performance of applying noise curriculum (NC) to other SSL models on ImageNet-100. Performance is improved for all tested models, **highlighting the broader applicability of our approach**. The accuracy curve over epochs are visualized in Figure 12 in Section B.5.

SSL Model	Architecture	Method	Accuracy
SimCLR	ResNet50	N2N + SimCLR	64.3
		SimCLR w/ NC	61.1
		SimCLR	59.0
MoCo v3	ViT-S	N2N + MoCo v3	60.4
		MoCo v3 w/ NC	55.3
		MoCo v3	52.2
SimSiam	ResNet50	N2N + SimSiam	68.4
		SimSiam w/ NC	65.7
		SimSiam	64.8
iBOT	ViT-S	N2N + iBOT	61.9
		iBOT w/ NC	62.7
		iBOT	56.9
DINO	ViT-S	N2N + DINO	62.5
		DINO w/ NC	62.1
		DINO	57.9
DINOv2	ViT-S	N2N + DINOv2	69.0
		DINOv2 w/ NC	68.1
		DINOv2	55.4

the frozen teacher and the DINOv2 backbone are derived from the same training run and pass through the same 30-epoch state, alignment is preserved. This allows us to meet the 100-epoch constraint while still leveraging a stronger teacher. Table 1b shows that **DINOv2 w/ NCT** improves over the **DINOv2** baseline with substantial absolute gains of 1.4% and 4.8% at noise levels $\sigma = 100$ and $\sigma = 255$, respectively. **DINOv2 w/ NC** also achieves a strong improvement of 2.5% under the more challenging $\sigma = 255$ setting. The relatively smaller gain of DINOv2 w/ NC at $\sigma = 100$ is due to the fact that, in large-scale settings with moderate noise, longer training alone significantly improves DINOv2’s robustness. Since DINOv2 w/ NC allocates a portion of the training budget to denoised pretraining, it shortens the noisy training phase, slightly limiting its benefit. Overall, while scaling training on large datasets improves DINOv2’s noise adaptation under moderate noise, denoised pretraining and regularization remain essential under high noise to maintain strong performance. **As an additional baseline**, we also evaluate DINOv2 trained exclusively on clean images and directly tested on noisy inputs. The results presented in Section C.1 confirm that clean-pretrained model suffers severe accuracy drop, which underscores the importance of our noise-aware training.

4.4 Probing on Clean Test Set Highlights Better Representation Quality

In previous sections, linear classification accuracies were measured on validation sets that had the same noise distribution as the training set. To further assess representation quality, we evaluate performance on the clean, original ImageNet validation set. Note that N2N + DINOv2 is evaluated without the denoiser because the clean images are already noise-free. Performance on this clean validation set serves as a strong indicator of how well a model captures representations of the "noise-free" real world. **Surprisingly, as highlighted by bold numbers in Table 2, DINOv2 w/ NC or NCT outperforms N2N + DINOv2 in most cases**, strongly suggesting that our denoiser-free approach learns more robust and generalizable representations than the two-stage denoising pipeline. This can be attributed to the fact that explicit denoising in the two-stage approach inevitably loses some useful information, leading to a degradation in representation quality in the second stage. This finding has important real-world implications: in scenarios where large-scale noisy data is readily available for pretraining, but only a limited amount of clean data is available for specific downstream tasks, DINOv2 w/ NC and NCT offer more effective solutions for learning transferable representations.

4.5 Instance Recognition Validates Versatility

To further validate the versatility of our method, we extend the evaluation to another downstream task. Here, we examine the instance-level recognition task on noisy images using the embeddings output by the backbone. The approach is non-parametric as the images’ embeddings are ranked based on their

Table 4: Mean Average Precision (mAP) on instance-level recognition tasks using Oxford and Paris datasets. **DINOv2 w/ NC** consistently outperforms **DINOv2** and closely matches **N2N + DINOv2** across benchmarks, with notable gains in extreme noise scenarios using regularization. **This validates the broader applicability of our approach to tasks beyond classification.**

Noise	Method	Oxford Paris		Noise	Method	Oxford Paris		Noise	Method	Oxford Paris	
Gaussian $\sigma=50$	N2N + DINOv2	20.89	38.62	Shot $\lambda=10$	N2N + DINOv2	21.36	40.00	Speckle $\sigma=102$	N2N + DINOv2	22.01	40.47
	DINOv2 w/ NC	21.30	40.94		DINOv2 w/ NC	21.94	42.20		DINOv2 w/ NC	22.40	41.28
	DINOv2	19.15	40.04		DINOv2	21.13	39.73		DINOv2	20.33	39.04
Gaussian $\sigma=100$	N2N + DINOv2	21.83	39.04	Shot $\lambda=3$	N2N + DINOv2	21.40	40.39	Speckle $\sigma=178.5$	N2N + DINOv2	21.31	39.31
	DINOv2 w/ NC	20.13	40.14		DINOv2 w/ NC	22.11	39.12		DINOv2 w/ NC	21.21	38.74
	DINOv2	16.39	32.54		DINOv2	18.59	35.65		DINOv2	18.63	37.06
Gaussian $\sigma=255$	N2N + DINOv2	15.67	33.68	Shot $\lambda=1$	N2N + DINOv2	18.96	37.22	Speckle $\sigma=255$	N2N + DINOv2	20.11	39.19
	DINOv2 w/ NCT	17.07	33.73		DINOv2 w/ NCT	20.66	36.57		DINOv2 w/ NCT	20.81	39.31
	DINOv2 w/ NC	14.81	29.97		DINOv2 w/ NC	19.25	36.67		DINOv2 w/ NC	21.10	38.59
	DINOv2	7.37	18.51		DINOv2	16.37	32.60		DINOv2	20.11	35.63
(a) Gaussian Noise				(b) Shot Noise				(c) Speckle Noise			

cosine similarity with a query image’s embedding. The results are based on the medium difficulty evaluation setup in the Oxford and Paris datasets, where both easy and hard images are treated as positive images. We measure mean average precision (mAP) and report results in Table 4. We see that the results closely align with Section 4.2. **DINOv2 w/ NC** greatly outperforms **DINOv2** in all comparisons. The mAP of **DINOv2 w/ NC** is on par with **N2N + DINOv2**, even at extreme noise levels. For example, at Shot $\lambda = 1$ and Speckle $\sigma = 255$, the mAP of **DINOv2 w/ NC** surpasses that of **N2N + DINOv2** on the Oxford dataset. **DINOv2 w/ NCT** outperforms all other methods the majority of the time in high noise settings, again validating its effectiveness.

4.6 DINOv2 w/ NC is Applicable to Diverse SSL Models

To test the applicability of DINOv2 w/ NC beyond DINOv2, we extend the DINOv2 w/ NC to various other SSL models on ImageNet-100, including SimCLR [9], MoCo v3 [11], SimSiam [10], iBOT [47], and DINO [7]. These models encompass both contrastive and non-contrastive approaches, as well as ViT-based [13] and CNN-based architectures [18]. All experiments are conducted with Gaussian ($\sigma=100$) noise without denoised regularization. As shown in Table 3, DINOv2 w/ NC improves over the noisy baseline across all models. The improvement is modest or minor for the first three models, i.e., SimCLR, MoCo v3, and SimSiam. The variation in contrastivity and backbone architecture among these models suggests that neither factor is critical to the method’s effectiveness. In contrast, iBOT and DINO (both are precursors to DINOv2) show substantial gains, similar to the improvements observed in DINOv2, where the **SSL Model w/ NC** closely matches **N2N + SSL Model**. This indicates that DINOv2 w/ NC has broad applicability, with more pronounced benefits for models sharing architectural and loss-function similarities with DINOv2. We also benchmark the performance of these models on the instance recognition task in Table 8 of Section B.5, which shows exactly identical trends as the linear evaluation, further reaffirming the generality of our approach. Detailed training configurations are provided in the Section A.4.

5 Limitations, Conclusion and Future Work

In this work, we proposed a fully self-supervised framework for learning noise-robust visual representations. Leveraging an SSL denoiser and the curriculum training strategy enables DINOv2 to improve significantly upon the noisy baseline. While our benchmark focuses on synthetic noise due to the scarcity of large-scale *labeled noisy* datasets that allows for standard downstream evaluation, this choice enables controlled, reproducible assessment across noise types. One limitation is the assumption that self-supervised denoiser can denoise reasonably well; while it is a fair assumption as we show in Section B.6 that even a very weak denoiser (1 training epoch) can still substantially improve accuracy, it remains an open dependency that could affect performance in more challenging settings. Another key direction for future research is to develop automated, adaptive strategies for curriculum schedule, as the current design requires tuning to determine when to shift from denoised to noisy samples during pretraining. More broadly, our methodology can be extended to other modalities like time series data, including noisy audio, EEG signals and financial data; we aim to establish a unified SSL approach for robust representation learning.

References

- [1] Mahmoud Assran, Randall Balestriero, Quentin Duval, Florian Bordes, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, and Nicolas Ballas. The hidden uniform cluster prior in self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2023.
- [2] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: a framework for self-supervised learning of speech representations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- [3] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari S. Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *ArXiv*, abs/2304.12210, 2023. URL <https://api.semanticscholar.org/CorpusID:258298825>.
- [4] Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. In *International Conference on Machine Learning*, 2019.
- [5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [6] Florent Brondolo and Samuel Beaussant. Dinov2 rocks geological image analysis: Classification, segmentation, and interpretability, 07 2024.
- [7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. doi: 10.1109/ICCV48922.2021.00951.
- [8] Shiyen Chen, Jiyuan Zhang, Zhaofei Yu, and Tiejun Huang. Exploring efficient asymmetric blind-spots for self-supervised denoising in real-world scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2814–2823, 2024.
- [9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.
- [10] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.
- [11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9640–9649, 2021.
- [12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- [13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>
- [14] European Space Agency (ESA). Sentinel-1 satellite data. <https://dataspace.copernicus.eu/explore-data/data-collections/sentinel-data/sentinel-1>, 2014.

- [15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.
- [16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Conference on Neural Information Processing Systems*, 2020.
- [17] Yue Guo, Shiqi Chen, Ronghui Zhan, Wei Wang, and Jun Zhang. Deformable feature fusion and accurate anchors prediction for lightweight sar ship detector based on dynamic hierarchical model pruning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 18:15019–15036, 2025. doi: 10.1109/JSTARS.2025.3574184.
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [19] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.
- [20] Tao Huang, Songjiang Li, Xu Jia, Huchuan Lu, and Jianzhuang Liu. Neighbor2neighbor: Self-supervised denoising from single noisy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14781–14790, June 2021.
- [21] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2129–2137, 2019.
- [22] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. High-quality self-supervised deep image denoising. In *Advances in Neural Information Processing Systems*, 2019.
- [23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. URL <https://api.semanticscholar.org/CorpusID:14542261>
- [24] Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. Ap-bsn: Self-supervised denoising for real-world images via asymmetric pd and blind-spot network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [25] Seong-Gyun Leem, Daniel Fulford, Jukka-Pekka Onnela, David Gard, and Carlos Busso. Adapting a self-supervised speech representation for noisy speech emotion recognition by using contrastive teacher-student learning. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10097135.
- [26] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. In *International Conference on Machine Learning*, 2018.
- [27] Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maass. Lodopab-ct, a benchmark dataset for low-dose computed tomography reconstruction. *Scientific Data*, 8(1): 109, 2021.
- [28] Cynthia McCollough, Adam Bartley, Rickey Carter, Baiyu Chen, Tammy Drees, Phillip Edwards, David Holmes III, Alice Huang, Farhana Khan, Shuai Leng, Kyle McMillan, Gregory Michalak, Kristina Nunez, Lifeng Yu, and Joel Fletcher. Low-dose ct for the detection and classification of metastatic liver lesions: Results of the 2016 low dose ct grand challenge. *Medical Physics*, 44:e339–e352, 10 2017. doi: 10.1002/mp.12345.

- [29] Omkar Nabar and Gautam Shroff. Conservative predictions on noisy financial data. In *Proceedings of the Fourth ACM International Conference on AI in Finance, ICAIF '23*, page 427–435, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702402. doi: 10.1145/3604237.3626859. URL <https://doi.org/10.1145/3604237.3626859>
- [30] Reiko Nakajima, R. Mandelbaum, U. Seljak, J. Cohn, Reinabelle Reyes, and Richard Cool. Photometric redshift requirements for lens galaxies in galaxy-galaxy lensing analyses. *Monthly Notices of the Royal Astronomical Society*, 420, 07 2011. doi: 10.1111/j.1365-2966.2011.20249.x.
- [31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.
- [32] Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. Recorrupted-to-recorrupted: Unsupervised deep learning for image denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2043–2052, June 2021.
- [33] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [34] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III* 18, pages 234–241. Springer, 2015.
- [36] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS' 16*, page 3637–3645, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.
- [37] Huy V. Vo, Vasil Khalidov, Timothée Darcet, Théo Moutakanni, Nikita Smetanin, Marc Szafraniec, Hugo Touvron, Camille Couprie, Maxime Oquab, Armand Joulin, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Automatic data curation for self-supervised learning: A clustering-based approach. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [38] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.
- [39] Yiming Wang, Jinyu Li, Heming Wang, Yao Qian, Chengyi Wang, and Yu Wu. Wav2vec-switch: Contrastive learning from original-noisy speech pairs for robust speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2022.
- [40] Zejin Wang, Jiazheng Liu, Guoqing Li, and Hua Han. Blind2unblind: Self-supervised image denoising with visible blind spots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2036, June 2022.
- [41] Marijke Welvaert and Yves Rosseel. On the definition of signal-to-noise ratio and contrast-to-noise ratio for fmri data. *PloS one*, 8:e77089, 11 2013. doi: 10.1371/journal.pone.0077089.
- [42] Xiaohu Wu, Ming Liu, Yue Cao, Dongwei Ren, and Wangmeng Zuo. Unpaired learning of deep image denoising. In *European Conference on Computer Vision (ECCV)*, 2020.
- [43] Huidong Xie, Weijie Gan, Bo Zhou, Ming-Kai Chen, Michal Kulon, Annemarie Boustani, Xiongchao Chen, Qiong Liu, Xueqi Guo, Menghua Xia, Liang Guo, Hongyu An, Ulugbek Kamilov, Hanzhong Wang, Biao Li, Axel Rominger, Kuangyu Shi, Ge Wang, and Chi Liu. Dose-aware diffusion model for 3d low-dose pet denoising: A multi-institutional validation with reader study and real low-dose data. *Journal of Nuclear Medicine*, 65, 2024.

- [44] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, 2021.
- [45] Dan Zhang, Fangfang Zhou, Felix Albu, Yuanzhou Wei, Xiao Yang, Yuan Gu, and Qiang Li. Unleashing the power of self-supervised image denoising: A comprehensive review, 2024. URL <https://arxiv.org/abs/2308.00247>
- [46] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.
- [47] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.
- [48] Shuang Zhou, Daochen Zha, Xiao Shen, Xiao Huang, Rui Zhang, and Fu-Lai Chung. Denoising-aware contrastive learning for noisy time series. In *International Joint Conferences on Artificial Intelligence*, 2024.
- [49] Qiu-Shi Zhu, Jie Zhang, Zi-Qiang Zhang, Ming-Hui Wu, Xin Fang, and Li-Rong Dai. A noise-robust self-supervised pre-training model based speech representation learning for automatic speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3174–3178. IEEE, 2022.

Outline of the Appendix:

- Section [A](#): Technical Details about Experiments
 - Section [A.1](#): Details of Toy Experiment in Section [3.2](#)
 - Section [A.2](#): Regularization Strength Details and Ablation
 - Section [A.3](#): Training Details for Main DINOv2 Experiments
 - Section [A.4](#): Training Details for Other SSL Models in Section [4.6](#)
 - Section [A.5](#): Noise Addition Formula
 - Section [A.6](#): Signal-to-Noise Ratio Details
- Section [B](#): Ablation Studies
 - Section [B.1](#): Ablation Unveils Synergy in Curriculum Learning
 - Section [B.2](#): Critical Roles of Alignment and Initialization
 - Section [B.3](#): Restart at Different Epochs
 - Section [B.4](#): Enduring Effectiveness in Mixed Noisy-Clean Data
 - Section [B.5](#): Further Results for Other SSL Models
 - Section [B.6](#): Robustness to Denoiser Quality
- Section [C](#): Baseline Comparisons
 - Section [C.1](#): Performance of Clean-Pretrained Model on Noisy Data
 - Section [C.2](#): Baseline Performance on Mixed Noisy-Denoised Data
- Section [D](#): Visualizations
 - Section [D.1](#): Training Loss Visualization
 - Section [D.2](#): Noisy Images Visualization
 - Section [D.3](#): PCA Visualization

A Technical Details about Experiments

A.1 Details of Toy Experiment in Section [3.2](#)

In both pretraining and linear probing evaluation, we use Adam optimizer, batch size 256, and a constant learning rate of 1.0×10^{-4} . The augmentation includes random resized crop and random affine, as shown in Listing [1](#). The strength λ for soft identity constraint is set to 0.5.

The results over 17 runs are visualized in Figure [6](#). We observe that adopting a noise curriculum not only significantly improves accuracy, **but also substantially stabilizes performance**, as evidenced by the narrower green box compared to the wider orange box. This highlights the dual benefits of curriculum learning in enhancing both robustness and consistency.

```
transforms.Compose([
    transforms.RandomResizedCrop(28, scale=(0.8, 1.0)),
    transforms.RandomAffine(
        degrees=15,
        translate=(0.1, 0.1),
        scale=(0.95, 1.05),
        shear=10
    ),
])
```

Listing 1: Data augmentation used in pretraining for the simple MLP with 1 hidden-layer in the toy experiment.

A.2 Regularization Strength Details and Ablation

For results in Figure [5](#), we select optimal regularization strength based on light sweeps visualized in Figure [7](#). Specifically, λ is set to 1.1, 5.0, 2.0 and 4.0 for Gaussian $\sigma = 255$, Shot $\lambda = 1$, Speckle $\sigma = 255$ and Gaussian $\sigma = 100$ respectively. Notably, Figure [7](#) shows that increasing the strength

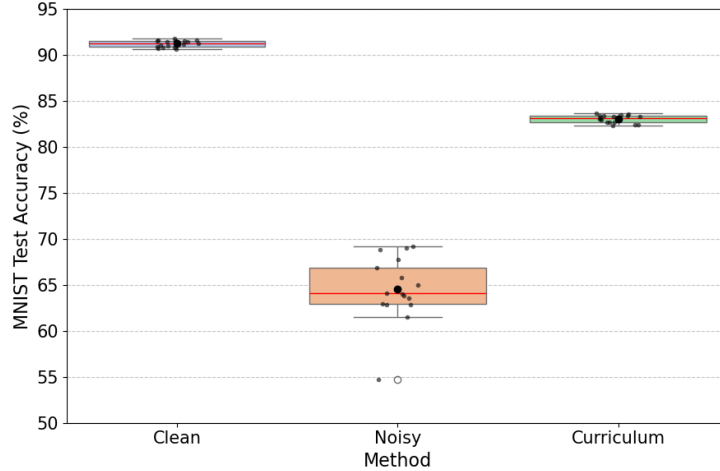


Figure 6: Box plots of toy experiments accuracies over 17 runs. "Clean" for MLP trained and tested on clean. "Noisy" for MLP trained and tested on noisy. "Curriculum" for MLP w/ NC tested on noisy. **Using noise curriculum not only significantly improves accuracy, but also stabilizes performance, as shown by the narrower green box compared to the wider orange box.**

results in a sharp increase in accuracy followed by fluctuations around a higher plateau, demonstrating the effectiveness of NCT.

In scaling epoch experiment for ImageNet-100 (Table 1a), λ is set to 1.1, 0.5 and 0.2 for 200, 500 and 1000 epochs respectively. We find longer training duration requires smaller regularization strength. This agrees with our main finding that scaling training can partially mitigate the performance degradation, thus requires less guidance from denoised teacher. In ImageNet-1k experiment (Table 1b), λ is set to 2.0 and 1.6 for Gaussian $\sigma = 100$ and Gaussian $\sigma = 255$.

For evaluation on clean set (Table 2), the strengths are set to the same value as their noisy counterparts detailed in the preceding paragraph.

In instance recognition experiments (Table 4), $\lambda = 0.2$ for all settings as we find it generally yields better results. For mixed noise-clean data evaluation shown later in Section B.4, λ is set to 1.1.

A.3 Training Details for Main DINOv2 Experiments

Training Details for Section 4.2 For curriculum-based training, although the model is trained on denoised images for only 140 epochs, the training dynamics in the first stage, such as learning rate scheduling, are configured based on the total training duration of 200 epochs. In the second stage of training on noisy images, the training dynamics are configured for a total duration of 60 epochs to ensure smooth convergence.

Training Details for Section 4.3 In ImageNet-1k experiments (Table 1b), the training configuration for DINOv2 and N2N + DINOv2 are set as default, consistent with its original implementation. In the noisy-stage training of DINOv2 w/ NC and DINOv2 w/ NCT, we reduce the linear learning rate warm-up epochs from 10 to 5, and the teacher temperature warm-up epochs from 30 to 15 to accommodate the reduced training duration for noisy adaptation under the constraint of 100 epochs. Intuitively, less warm-up is needed because denoised pre-training provides a good initialization.

Training Details for SimCLR Baseline in Table 1b Consistent with DINOv2, we set batch size to 512 and train for 100 epochs during pre-training. We use the open-source PyTorch implementation from [AndrewAtanov/simclr-pytorch](#). Since SimCLR typically requires more epochs during linear evaluation compared to DINOv2, we train the linear head for 90 epochs (112.59k steps) with a batch size of 1024 to ensure fair comparison.

Details of Linear Probing. Consistent with the original DINOv2 paper, when performing linear probing, a light grid search is conducted over hyperparameters, i.e., learning rate, output layer, and

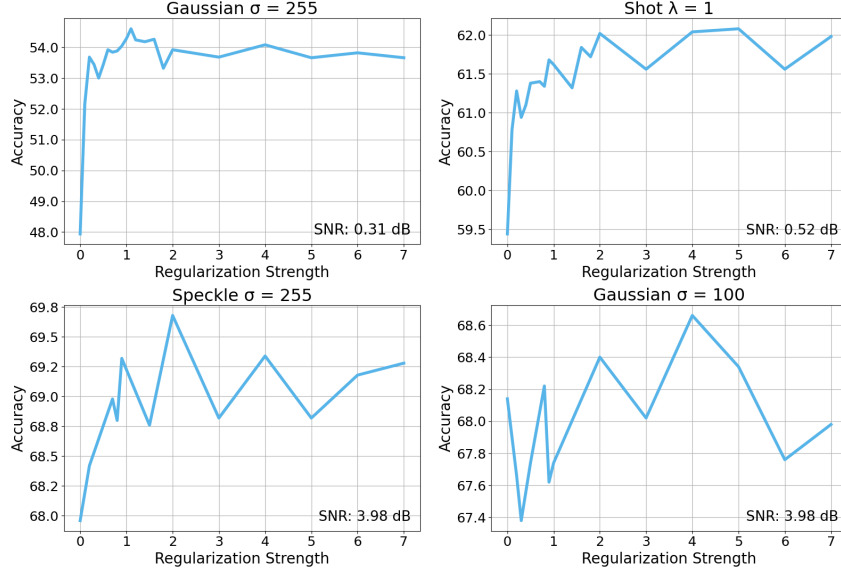


Figure 7: Regularization strength ablation to determine optimal strength for DINOv2 w/ NCT in Figure 5. Most plots show a **rapid increase in linear probing accuracy followed by fluctuations around a higher plateau as strength increases**. Due to computational constraints, the sweep was not performed with uniform resolution across all settings and may vary across noise types.

whether concatenating average-pooled tokens. The highest accuracy value is reported which is a common practice.

A.4 Training Details for Other SSL Models in Section 4.6

The linear evaluations of all models train the linear head for 50 epochs.

SimCLR. During training, we use a batch size of 100 and LARS optimizer. After square root scaling, the base learning rate is set to $0.075 \times \sqrt{100} = 0.75$. During linear evaluation, we use a batch size of 200 and SGD optimizer. After square root scaling, the base learning rate is set to $1.6 \times \sqrt{200/4096} = 0.35$.

MoCo v3. During training, we use a batch size of 128 and AdamW optimizer. After square root scaling, the base learning rate is set to $4 \times 10^{-4} \times \sqrt{128/1024} = 1.414 \times 10^{-4}$. During linear evaluation, we use a batch size of 200 and SGD optimizer. After linear scaling, the base learning rate is set to $0.2 * (200/256) = 0.15625$.

SimSiam. During training, we use a batch size of 100 and SGD optimizer. After square root scaling, the base learning rate is set to $0.15 \times \sqrt{100/256} = 0.09375$. During linear evaluation, we use a batch size of 200 and LARS optimizer. We found the learning rate provided in SimSiam’s codebase is too small for effectively training the linear head. After some experimentation, we set the base learning rate to 2.34375 after the linear scaling $3.0 \times (200/256)$.

iBOT. During training, we use a batch size of 80 and AdamW optimizer. After square root scaling, the base learning rate is set to $5 \times 10^{-4} \times \sqrt{80/256} = 2.795 \times 10^{-4}$. During linear evaluation, we use a batch size of 128. We also adopt the sweeping hyperparameter implementation in iBOT to obtain the best linear evaluation value. This involves sweeping over a range of learning rates, and a set of candidate optimizers including LARS and SGD.

DINO. During training, we use a batch size of 128 and AdamW optimizer. After square root scaling, the base learning rate is set to $5 \times 10^{-4} \times \sqrt{128/256} = 3.54 \times 10^{-4}$. During linear evaluation, we use a batch size of 128 and SGD optimizer. After square root scaling, the base learning rate is set to $0.001 \times \sqrt{128/256} = 7.1 \times 10^{-4}$.

A.5 Noise Addition Formula

As stated in Section 4.1, we follow ImageNet-C [19] to introduce Gaussian, Shot, and Speckle noises to the images. Let original image be x , and normalized image be $\tilde{x} = x/255$.

Gaussian noise:

$$x_{\text{noisy}} = 255 \cdot \text{clip}(\tilde{x} + \mathcal{N}(0, c^2), 0, 1) \quad (7)$$

where $c = \frac{\sigma}{255}$, and $\sigma \in \{50, 100, 255\}$.

Shot (Poisson) noise:

$$x_{\text{noisy}} = 255 \cdot \text{clip}\left(\frac{\text{Poisson}(\tilde{x} \cdot \lambda)}{\lambda}, 0, 1\right) \quad (8)$$

where $\lambda \in \{10, 3, 1\}$.

Speckle noise:

$$x_{\text{noisy}} = 255 \cdot \text{clip}(\tilde{x} + \tilde{x} \cdot \mathcal{N}(0, c^2), 0, 1) \quad (9)$$

where $c = \frac{\sigma}{255}$, and $\sigma \in \{102, 178.5, 255\}$.

Note that the clipping is applied to the $[0, 1]$ range in all noise types, which further increases the difficulty of denoising at high noise levels. A significant portion of the noise signal may get clipped and hence irreversibly distorted or lost.

A.6 Signal-to-Noise Ratio Details

All SNR values given are calculated from the raw noisy images without preprocessing, e.g., resizing, crop. We report the average SNR between the training set and the validation set since they have no significant difference. In the instance recognition, we report the average between the Oxford and Paris datasets. Detailed per-set SNR values are provided in Table 5 and Table 6.

Table 5: SNR values of training and validation sets for ImageNet-100 and ImageNet-1k datasets.

Dataset	Noise Type	Train SNR (dB)	Val SNR (dB)
ImageNet-100	Gaussian $\sigma = 50$	8.84	8.63
	Gaussian $\sigma = 100$	4.09	3.87
	Gaussian $\sigma = 255$	0.43	0.19
	Shot $\lambda = 10$	8.55	8.44
	Shot $\lambda = 3$	4.16	4.05
	Shot $\lambda = 1$	0.58	0.47
	Speckle $\sigma = 102$	9.17	9.14
	Speckle $\sigma = 178.5$	5.59	5.56
	Speckle $\sigma = 255$	4.00	3.96
ImageNet-1k	Gaussian $\sigma = 100$	4.41	4.31
	Gaussian $\sigma = 255$	0.78	0.67

Table 6: SNR values for Oxford and Paris datasets.

Noise Type	Oxford SNR (dB)	Paris SNR (dB)
Gaussian $\sigma = 50$	8.86	9.04
Gaussian $\sigma = 100$	4.10	4.27
Gaussian $\sigma = 255$	0.39	0.55
Shot $\lambda = 10$	8.66	8.79
Shot $\lambda = 3$	4.24	4.39
Shot $\lambda = 1$	0.65	0.78
Speckle $\sigma = 255$	4.09	4.17
Speckle $\sigma = 178.5$	5.69	5.75
Speckle $\sigma = 102$	9.28	9.31

B Ablation Studies

B.1 Ablation Unveils Synergy in Curriculum Learning

We conduct ablation studies on ImageNet-100 to analyze the contribution of the two key components of the curriculum learning method, specifically in the context of linear probing classification on images with Gaussian noise ($\sigma = 100$).

Only Denoised Pretraining. We train DINOv2 on denoised images for 140 epochs, then **continues** to train it on noisy images for 60 epochs without resetting training dynamics.

Only Noisy Training with Restart. We train DINOv2 on noisy images for 140 epochs, then **restart** training on the same noisy set for 60 epochs.

Figure 8 illustrates that denoised pretraining contributes nearly twice the accuracy improvement over the noisy baseline compared to restarting training. Notably, their effects are not merely additive, as our combined method outperforms the sum of their individual gains, highlighting a synergistic interaction between the two techniques. The synergy arises because, while denoised pretraining provides robust initial weights, the diminished learning rate in later stages limits adaptation to noisy images. Restarting training resets the learning rate, allowing the model to fully adapt to noise and leverage the learned representations effectively.

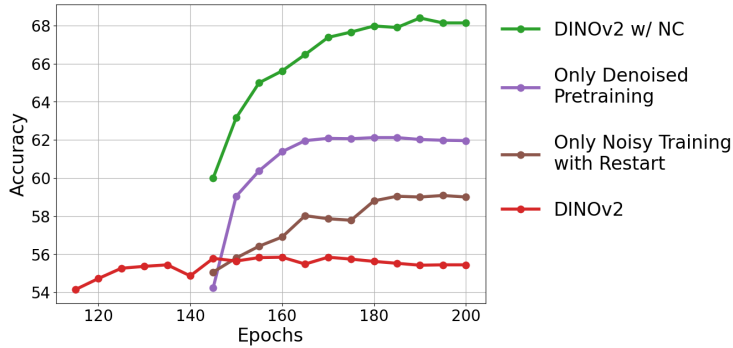


Figure 8: Comparison of curriculum learning ablations showing the contributions of denoised pretraining and restarting noisy training to linear evaluation accuracy. **“Only Denoised Pretraining” (purple) provides nearly double the improvement of “Only Noisy Training with Restart” (brown) over the Noisy baseline (red).**

B.2 Critical Roles of Alignment and Initialization

To evaluate the design choices of our regularization loss, we conduct ablation studies on ImageNet-100 under extreme Gaussian noise ($\sigma = 255$), where the regularization loss has the most significant impact. These studies provide insights into the importance of alignment and initialization in our approach.

Unaligned Frozen Teacher. Instead of extracting the weights of the current teacher backbone at the end of the denoised training phase, we retrain a separate backbone on the same denoised dataset to extract teacher weights. This alternative frozen teacher is then used during the noisy training phase. Unlike the original method, this introduces misaligned embeddings between the frozen teacher, the trainable teacher, and the student. At regularization strength 0.2, as shown by the brown line in Figure 9, this setup results in a marginal performance improvement of just 1.1% over the green baseline, significantly lower than the aligned case depicted by the purple line. At regularization strength 1.1, as shown in Figure 10, the strong unaligned teacher instead destabilizes training, dragging accuracy downward as learning proceeds, ultimately underperforming even the DINOv2 w/ NC. These results highlight the seamless integration between the regularization loss and the curriculum training pipeline, where the first stage’s outputs are essential for effectively regularizing the second stage.

Randomly Initialized Backbone. To further investigate the role of denoised pretraining, we replaced the trainable teacher and student backbones with randomly initialized weights during the noisy training phase while keeping the frozen teacher unchanged. At regularization strength 0.2, this configuration yields extremely poor performance, with accuracy dropping below 20%, as shown

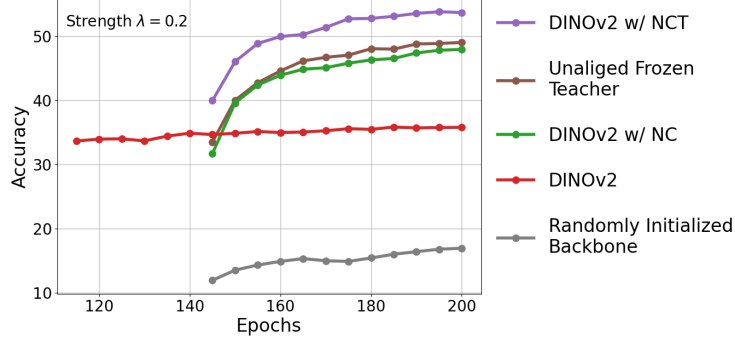


Figure 9: Comparison of regularization ablations at strength = 0.2 showing the contribution of alignment and initialization to linear evaluation accuracy. Aligned regularization (purple) significantly outperforms unaligned regularization (brown) and random initialization (grey). **This illustrates both the alignment and the denoised pretraining contributes substantially to the effectiveness of the regularization.**

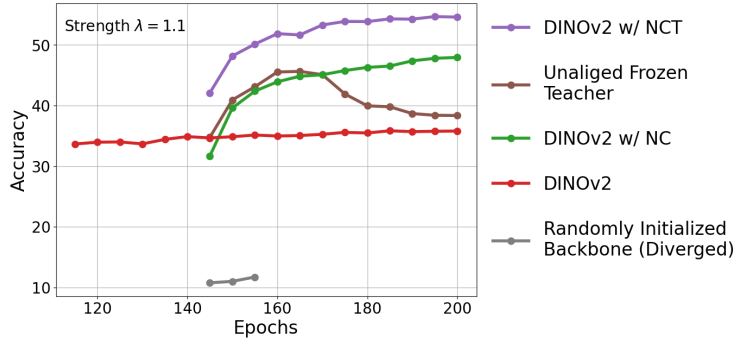


Figure 10: Comparison of regularization ablations at strength = 1.1 showing the contribution of alignment and initialization to linear evaluation accuracy. Aligned regularization (purple) significantly outperforms unaligned regularization (brown) and random initialization (grey). **Random initialization diverges after 30 epochs of training. Strong unaligned regularization drags the accuracy down as training progresses.**

by the gray line in Figure 9. Notably, at regularization strength 1.1, the training diverges after 30 epochs as the loss explodes, as shown by the discontinuous grey line in Figure 10. These results highlight the necessity of denoised pretraining, even in the presence of a regularization term, as it provides a crucial foundation for the training process. Without it, the training fails to leverage the frozen teacher’s guidance.

B.3 Restart at Different Epochs

We investigated the impact of restarting training on noisy data at different stages: (1) training on denoised data for 140 epochs, followed by 60 epochs on noisy data (a common setting in this paper); (2) 130 epochs on denoised data, then 70 epochs on noisy data; and (3) 120 epochs on denoised data, then 80 epochs on noisy data. As shown in Figure 11, the performances across these variations are similar, with no approach consistently outperforming the others. Thus, we decide to use 140 epochs as the restart point.

B.4 Enduring Effectiveness in Mixed Noisy-Clean Data

In real-world scenarios, datasets often contain images of varying quality, posing challenges for SSL methods. In this section, we investigate how different noise-clean distributions on ImageNet-100 affect the SSL model’s performance and the effectiveness of our method under these distributions. We replace $x\%$ (i.e., 0%, 2%, 10%) of noisy data with clean images. So in our noise curriculum, the model is trained on a mix of $(100 - x)\%$ denoised and $x\%$ clean images in the first stage, then on a mix of $(100 - x)\%$ noisy and $x\%$ clean images in the second stage. The validation set is also adjusted to follow the same distribution as the training set. From Table 7, we can see that generally

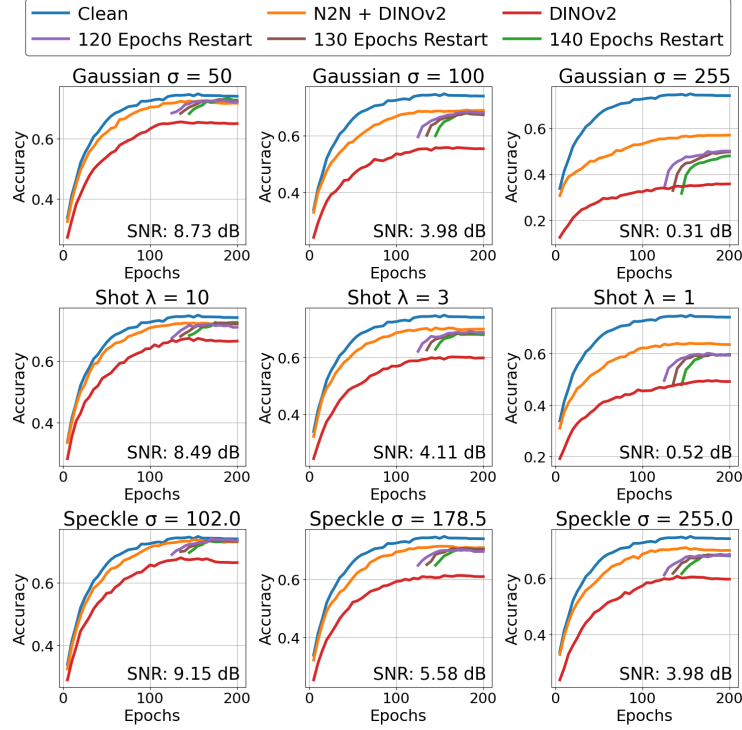


Figure 11: Linear Probing Classification accuracy for different restarting epochs. **We observe that no approach consistently outperforms the others between restarts at 120, 130 and 140 epochs.**

the linear evaluation accuracy decreases across all methods when more clean data is introduced but noisy data remains the majority, which can be attributed to a reduced overall representation alignment between noisy, denoised and actual clean data. On the other hand, through column-wise comparisons, **DINOv2 w/ NC** consistently outperforms **DINOv2** and closely matches **N2N + DINOv2**; regularized method **DINOv2 w/ NCT** brings substantial improvement over unregularized method. It is clear that our curriculum learning framework and denoised regularization are still effective despite the varying training distributions.

Table 7: Mixed Noise-Clean Data Evaluation: Impact of introducing varying percentages of clean data (0%, 2%, 10%) on linear evaluation accuracy. We observe that increasing the proportion of clean data generally reduces accuracy across all methods due to reduced representation alignment. **However, our DINOv2 w/ NC and NCT are still effective despite the varying training distributions.**

Noise Type	Method	0% Clean	2% Clean	10% Clean
Gaussian-50	N2N + DINOv2	71.8	72.0	71.1
	DINOv2 w/ NC	72.8	72.3	71.6
	DINOv2	65.0	61.2	61.9
	N2N + DINOv2	69.0	67.8	67.7
Gaussian-100	DINOv2 w/ NC	68.1	66.6	67.0
	DINOv2	55.4	54.8	53.1
	N2N + DINOv2	57.0	55.2	53.5
	DINOv2 w/ NCT	54.6	53.8	51.7
Gaussian-255	DINOv2 w/ NC	47.9	47.3	47.3
	DINOv2	35.8	36.9	35.8

Table 8: Mean Average Precision (mAP) on instance-level recognition tasks for other SSL models. The pattern is consistent with Table 3. Noise curriculum (NC) improves performance for all models.

SSL Model	Method	Oxford - M	Paris - M
SimCLR	N2N + SimCLR	21.43	35.32
	SimCLR w/ NC	19.08	33.39
	SimCLR	18.04	31.65
MoCo v3	N2N + MoCo v3	15.34	29.52
	MoCo v3 w/ NC	13.79	30.05
	MoCo v3	12.95	27.72
SimSiam	N2N + SimSiam	24.03	40.10
	SimSiam w/ NC	21.12	39.77
	SimSiam	20.64	38.15
iBOT	N2N + iBOT	12.26	25.52
	iBOT w/ NC	14.60	29.79
	iBOT	11.05	24.05
DINO	N2N + DINO	15.04	29.16
	DINO w/ NC	16.01	31.33
	DINO	12.73	27.70
DINOv2	N2N + DINOv2	21.83	39.04
	DINOv2 w/ NC	20.13	40.14
	DINOv2	16.39	32.54

B.5 Further Results for Other SSL Models

Figure 12 shows the accuracy over epochs for SSL models tested in Section 4.6. Table 8 shows the performance of the SSL models on the instance recognition task using the Oxford and Paris dataset. Both results illustrate that the noise curriculum improves the performance of all SSL models on noisy data, demonstrating the wide applicability of the method.

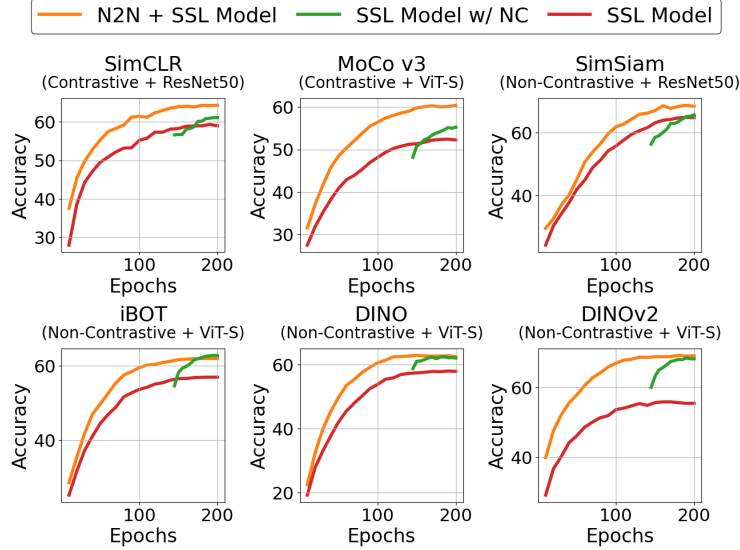


Figure 12: Linear evaluation performance over epochs of applying DINOv2 w/ NC to other SSL models on ImageNet-100. Noise curriculum (green) improves performance for all tested models, with substantial gains observed in iBOT and DINO variants, showing similar trend as DINOv2.

B.6 Robustness to Denoiser Quality

To investigate the level of tolerance of DINOv2 w/ NC to the denoiser quality, we conduct ablation studies on how the number of denoiser training epochs impacts the SSL model’s final performance. We use the number of training epochs as a quantifiable proxy for denoiser effectiveness. The evaluation follows our default setting: 200 epochs of DINOv2 ViT-S training with Gaussian-100 noisy data. The results are shown in Table 9. We observe that even training the N2N denoiser for just 1 epoch leads to a substantial improvement of DINOv2 w/ NC over the DINOv2 baseline (i.e., 10.7 increase), while being only 2 percent less than that of a well-trained denoiser (100 epochs). Although the 1-epoch denoiser still produces many undesirable artifacts like a very weak denoiser, the curriculum still yields substantial improvements in downstream performance. This highlights the robustness of our curriculum, which has considerable tolerance to the denoiser’s quality. A denoiser has to be unreasonably bad (e.g., training diverged or collapsed) to provide no return in performance.

Table 9: Performance of DINOv2 w/ NC on ImageNet-100 with the N2N denoiser trained for varying epochs. Notably, even a weak denoiser trained for only 1 epoch substantially improves DINOv2 w/ NC over DINOv2.

Method	Denoiser Training Epochs	Accuracy
DINOv2	/	55.4
DINOv2 w/ NC	1 epoch	66.1
DINOv2 w/ NC	5 epochs	67.4
DINOv2 w/ NC	100 epochs	68.1

Table 10: Performance of clean-pretrained DINOv2 models under different noise conditions. Linear probes are fine-tuned either on clean or noisy data while keeping the backbone frozen. **The clean-pretrained DINOv2 suffers severe drop in performance when applied on the noisy test set.**

Training Dataset	Architecture	Eval Dataset	Eval Noise	Probe Tuned on Clean	Probe Tuned on Noisy
LVD-142M	ViT-B/14	ImageNet-1k	Gaussian-100	63.2	70.3
			Gaussian-255	8.9	29.0
			Clean	84.5	/
LVD-142M	ViT-S/14	ImageNet-1k	Gaussian-100	49.2	61.0
			Gaussian-255	4.9	23.1
			Clean	81.1	/
ImageNet-1k (100 epochs)	ViT-B/16	ImageNet-1k	Gaussian-100	40.7	58.1
			Gaussian-255	2.2	20.5
			Clean	79.0	/
ImageNet-100 (1000 epochs)	ViT-S/16	ImageNet-100	Gaussian-50	60.0	74.3
			Gaussian-100	22.3	61.3
			Gaussian-255	3.1	34.5
			Clean	81.4	/

Table 11: Linear probing accuracy on noisy ImageNet-100 test set when training on mixed noisy+denoised data. Mixing denoised images results in a 7.4 point drop compared to training on noisy data alone.

Training Setting	Accuracy
100% Gaussian-100	55.4
50% Gaussian-100 + 50% Denoised	48.0

C Baseline Comparisons

C.1 Performance of Clean-Pretrained Model on Noisy Data

We evaluate the noise-robustness of a DINOv2 model trained exclusively on clean images, in order to reinforce the necessity of noisy training. Two strategies are employed to cover complementary baselines. (1) With the backbone frozen, we fine-tune a linear probe on noisy data drawn from the same distribution as the test set, and then evaluate the full model on noisy data. (2) With the backbone again frozen, we fine-tune a linear probe on clean data and subsequently evaluate the model on noisy data. The first strategy simulates a common downstream scenario where a frozen feature extractor is adapted to a new domain via task-specific heads, while the second serves as a more standard adversarial benchmark that tests the model’s robustness to noise without ever exposing it to noise during training. In addition to our own trained models, we also benchmark the officially released DINOv2 weights, which were pretrained on the LVD-142M dataset (142 million images). The results in Table 10 show substantial performance drops (15–70 points) when applying clean-pretrained DINOv2 to noisy test sets. The degradation is particularly severe on Gaussian-255 noise, where the accuracy of ViT-B/14 drops from 84.5 to 29.0 (a 55.5-point decrease), even when the probe is tuned on Gaussian-255 noisy data.

Notably, despite the smaller model size and significantly reduced training data, our DINOv2 w/ NCT (Table 1b) surpasses the official ViT-B/14 weights trained on LVD-142M at both Gaussian-100 (72.1 vs. 70.3) and Gaussian-255 (55.8 vs. 29.0). These results emphasize that standard DINOv2 is not robust to input noise, underscoring the necessity of incorporating noisy images during pretraining to achieve strong downstream performance in noisy settings.

C.2 Baseline Performance on Mixed Noisy–denoised Data

Another intuitive baseline is to investigate whether mixing denoised samples with noisy ones during training can improve representation learning by increasing the diversity and expressiveness of inputs. Table 11 shows the linear probing accuracies of DINOv2 ViT-S on the noisy ImageNet-100 test set,

when trained for 200 epochs on 100% noisy data (from Table 1(a)) versus on 50% noisy + 50% denoised data. We observe a significant drop of 7.4 points in accuracy when noisy and denoised data are randomly mixed during training. As a result, mixing clean (denoised) and noisy images during training leads to degraded performance, primarily due to reduced representation alignment between noisy and denoised samples, which hinders the model’s ability to learn consistent features. This shows naive mixing is suboptimal, and reinforces the necessity of using a staged curriculum learning approach that separates noisy and denoised images.

D Visualizations

D.1 Training Loss Visualization

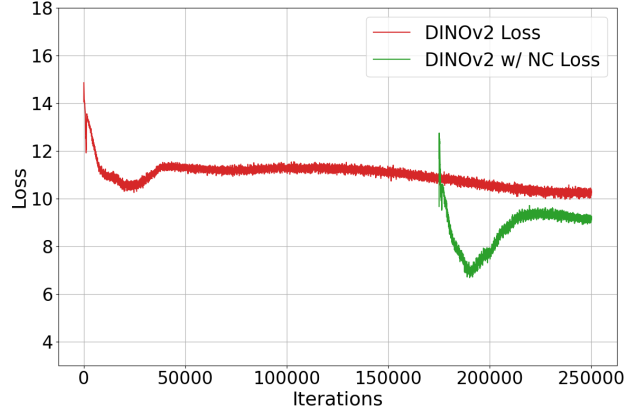
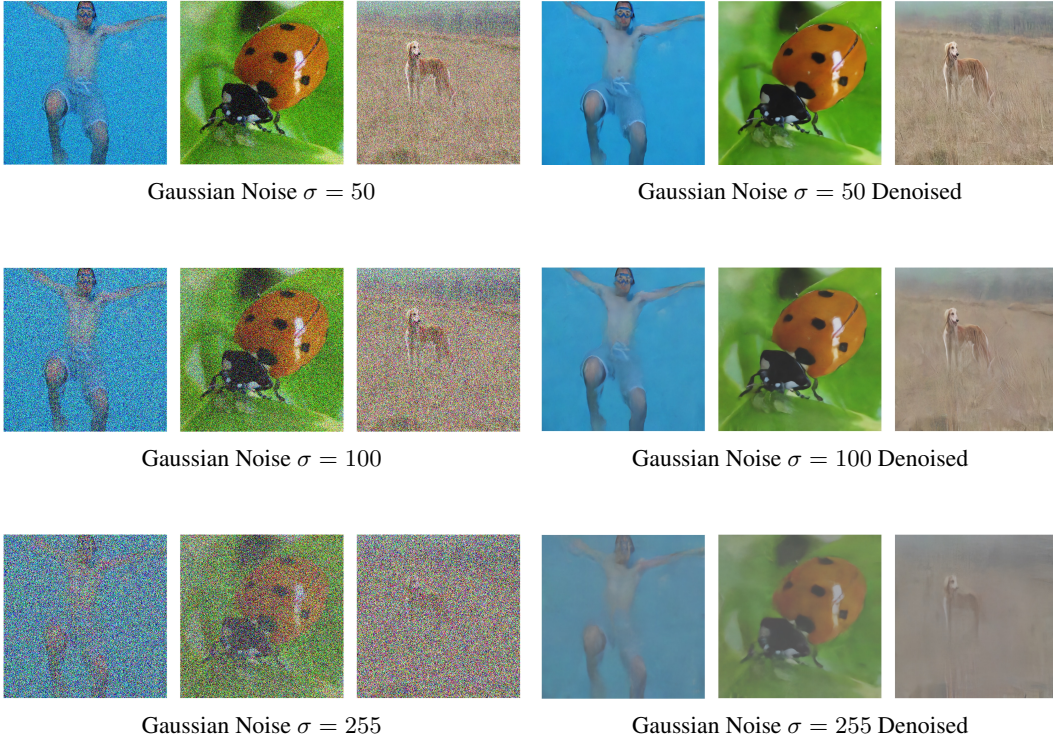
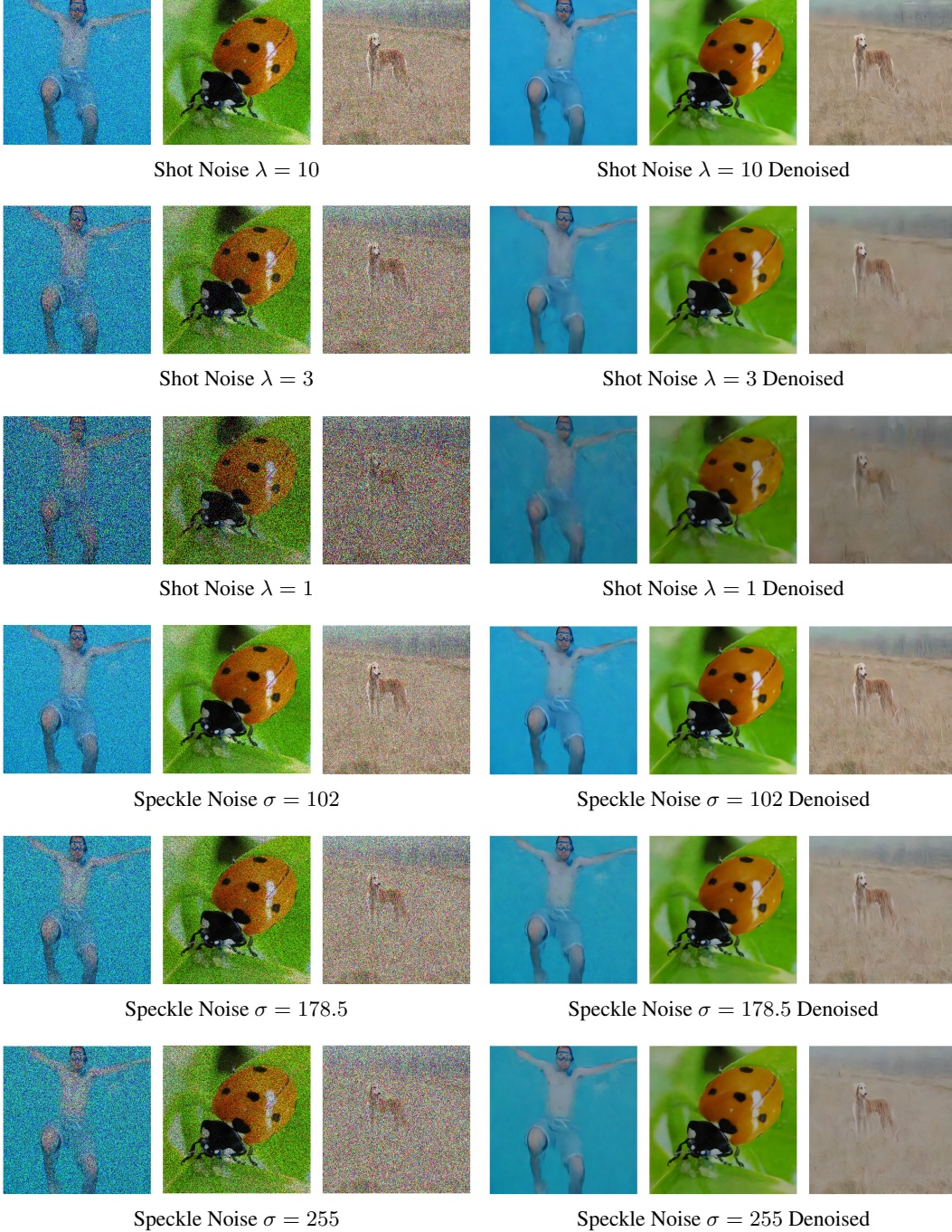


Figure 13: Comparison of training loss between DINOv2 and DINOv2 w/ NC. **DINOv2 w/ NC can descend to a lower loss due to its better initialization from the denoised pretraining.**

Figure 13 shows the comparison between training loss of DINOv2 and DINOv2 w/ NC on the ImageNet-100 noisy data with Gaussian noise ($\sigma = 100$). DINOv2 w/ NC loss (green) descends to a lower value than DINOv2 (red). The two curves are similar in shape which reflects the restarting training technique in our noise curriculum learning.

D.2 Noisy Images Visualization





D.3 PCA Visualization

To visualize the dense features, we apply PCA to the feature space and map the first three principal components to the RGB channels. We use the ViT-S 200-epoch Gaussian-255 checkpoints from Table 1a for visualization. As shown in , the DINOv2 visualizations (middle of each triplet) fail to clearly separate salient objects from the background, and the resulting color patterns lack semantic consistency across images, suggesting weaker representation quality. In contrast, the DINOv2 w/ NCT visualizations (right of each triplet) distinctly separate salient objects, and objects of the same class are consistently represented with similar color patterns across different images, indicating semantically aligned features. These results demonstrate that NCT yields a substantial improvement in feature quality over the DINOv2 baseline.

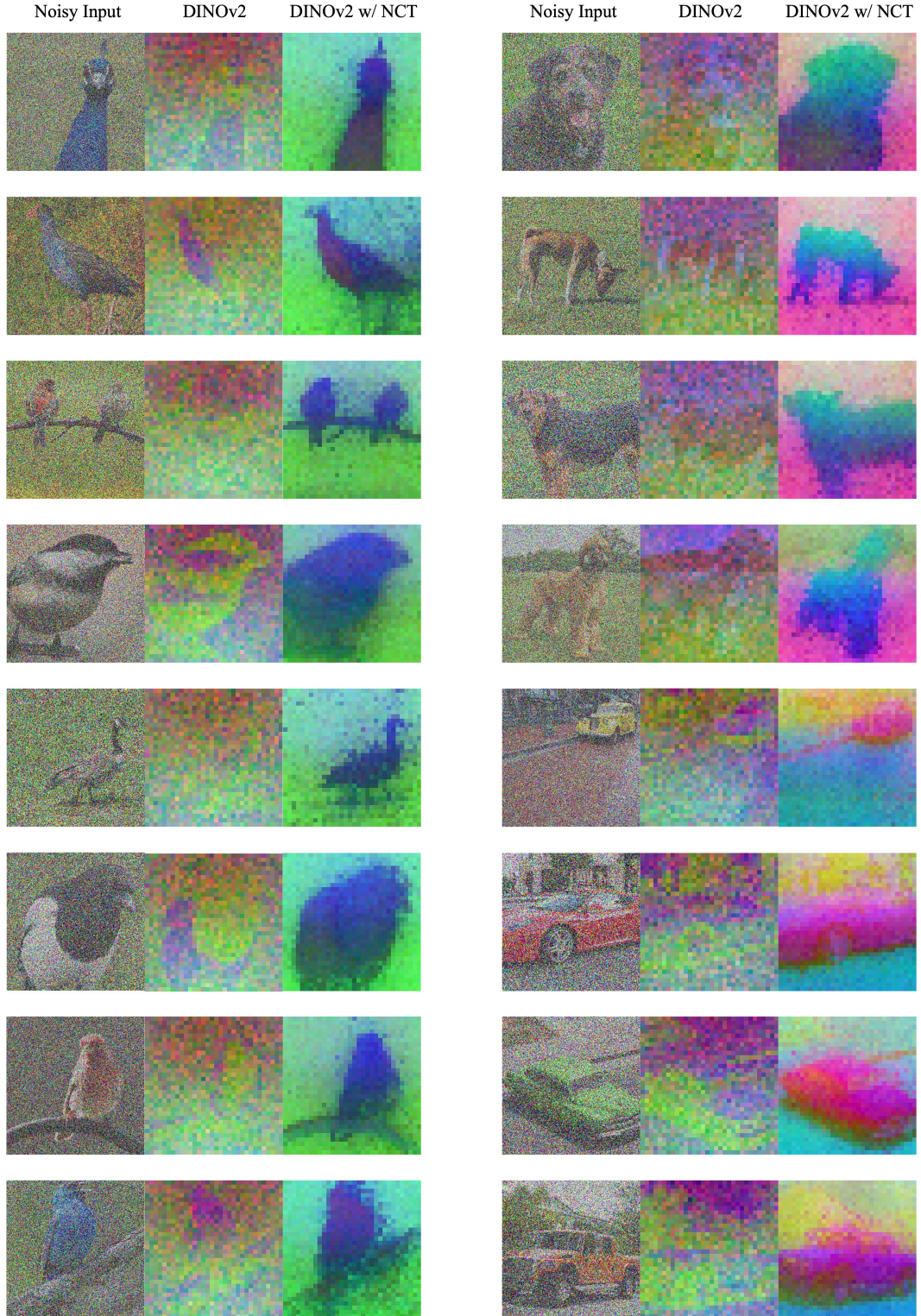


Figure 23: PCA visualizations: we visualize dense features of the models by performing PCA on the feature space and mapping the top three principal components to RGB. In each triplet, the left image is the noisy input with Gaussian noise at $\sigma = 255$. The middle and right images are produced by DINOv2 and DINOv2 w/ NCT (ViT-S, 200 epochs) respectively.