

Learning Beyond the Training Distribution

Distributional and Structural Extrapolation

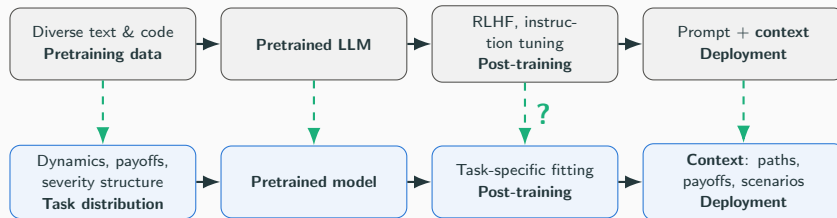
Ali Hasan

September 2026

Thank you to collaborators

- Waiss Azizian
- Patrick Kuiper
- Yuting Ng
- Agostino Capponi
- Anderson Schneider

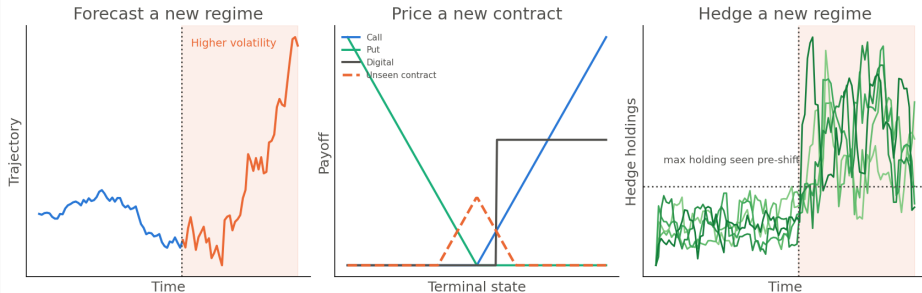
Foundation Model Paradigm



Motivating Questions:

1. How do we obtain extrapolation, architecturally?
2. What does the pretrained prior actually cover?

Foundation Models for Finance?

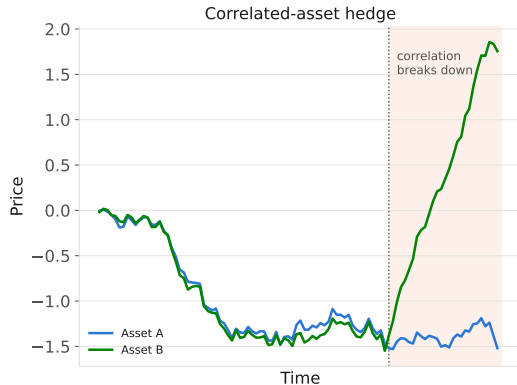


How do we build a pretrained model that extrapolates to regions of interest in numerical and financial domains?

Why does this matter more here than for language?

LLM-generated code

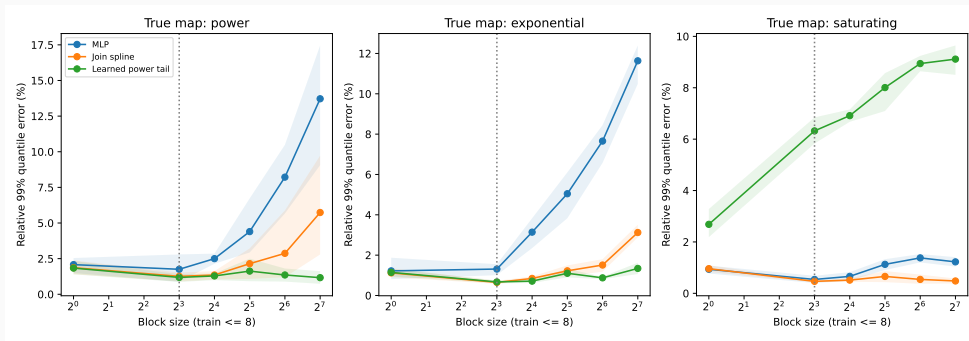
```
def value_at_risk(returns, alpha=0.01):  
    mu = returns.mean()  
    sigma = returns.std()  
    return norm.ppf(alpha, mu, sigma)
```



Language has coherent structure almost everywhere you'd sample it. Numerical data of interest can quickly move outside the convex hull of the pretraining mix.

Example 1: the architecture decides what extrapolates

Consider a max-stable latent variable. Train on blocks ≤ 8 ; test to $16\times$ that:



Indistinguishable in range. Out of range, matching the growth class cuts error $\sim 10\times$ when the truth is power-law or exponential except when output is bounded.

Related Work and Motivation

Part II on pretraining relies on **in-context learning**: a frozen model, given a context of examples, predicts the next one, with no weight updates.

$$\text{Context} : (x_1, y_1), \dots, (x_n, y_n) \quad \text{Query} : (x_{n+1}, ?)$$

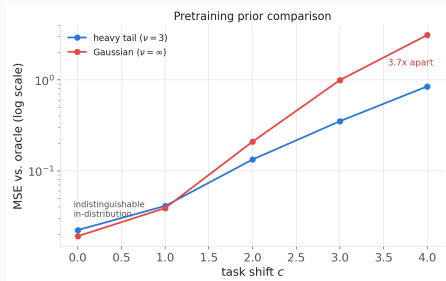
Adapting by context only works if it does more than recall seen tasks:

- **von Oswald et al., 2023**: in-context learning can implement gradient descent, other algorithms. If it learns an algorithm, can it generalize OOD?
- **Chan et al., 2022; Raventos et al., 2023**: heavy-tailed pretraining data promotes the emergence of in-context learning in the first place.
- **Hollmann et al., 2023**: for tabular foundation models, the pretraining distribution is a choice with tradeoffs. What tradeoffs are we okay with?

Why it matters: if ICL is more than recall, can we use this to build models that extrapolate?

Example 2: the pretraining prior decides what extrapolates

Take a prediction example of an Ornstein-Uhlenbeck process: only the prior over pretraining tasks differs, heavy-tailed Student- t versus Gaussian (i.e. $k = \infty$):



Both perform reasonably in-distribution, but error grows $3.7\times$ apart the moment the test task moves away from the pretraining bulk.

This talk

We describe how extrapolation trades in-distribution fit for extrapolating capability.

Architectural Invariances

A configuration more extreme than anything observed, in the same regime.

Part I: max-stable representations – addressed through architecture.

Pretraining Distribution

A task or dynamics unlike the pretraining distribution.

Part II: pretraining priors for ICL – addressed through the prior.

Practical Impact

Pricing new contracts, zero-shot: can we get variance reduction across many stochastic processes, without hand-deriving an estimator each time?

Part III: Example applications in Monte Carlo.

Future Studies

Part IV: Unifying directions.

Part I: Max-Stable Representations

Goal of Part I

We need a way to define extremeness and extrapolation in high dimensional spaces.

- We want to define a new notion of extremeness which is relevant for our tasks (since coordinatewise max may not be appropriate)
- We want to endow it with known extrapolating statistical behavior (extreme value theory)

Background: Extreme Value Theory (EVT)

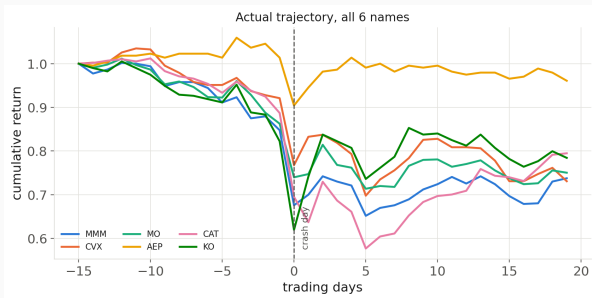
Let X_i be i.i.d random variables $\sim F$. We are interested in characterizing the behavior of maxima that's the distribution of $M_n = \max_{1 \leq i \leq n} X_i$. Then, similar to the central limit theorem, it is possible to find a location and scale b_n and a_n s.t.

$\mathbb{P}[(M_n - b_n)/a_n \leq x] \rightarrow H(x)$ where $H(x)$ is non-degenerate.

$$H_{\xi}(x) = \begin{cases} \exp(-(1 + \xi x)^{-1/\xi}), & \text{if } \xi \neq 0 \\ \exp(-e^{-x}), & \text{if } \xi = 0 \end{cases} \quad (1)$$

- $\xi < 0$: the Weibull distribution (short-tailed).
- $\xi = 0$: The Gumbel distribution (tail decays exponentially).
- $\xi > 0$: The Fréchet distribution (heavy-tailed).

Example on crashes



Six S&P 500 names, 1964–2024. Worst joint 5-day window in the sample on Black Monday, 1987. Compute probabilities for two models fit on the other 59.9 years:

	Gaussian	GPD
MMM	$6.2 \times 10^{42} \text{ yr}$	771 yr
CVX	$9.1 \times 10^{21} \text{ yr}$	127 yr
KO	$2.9 \times 10^{64} \text{ yr}$	1,100 yr

Implied return period. It happened once in 60 years.

Background: Max-Stability

A distribution with location a and scale b denoted by $P_{\mu,\sigma}$ is *max-stable* if for iid samples $\{X^{(i)}\}_{i=1}^N \sim P_{\mu,\sigma}$, there exists a sequence a_N, b_N such that for all N

$$a_N \max_{i=1}^N X^{(i)} + b_N \sim P_{\mu,\sigma}.$$

The maximum of iid samples from a distribution is the same distribution up to an affine transformation!

Background: EVT vs CLT

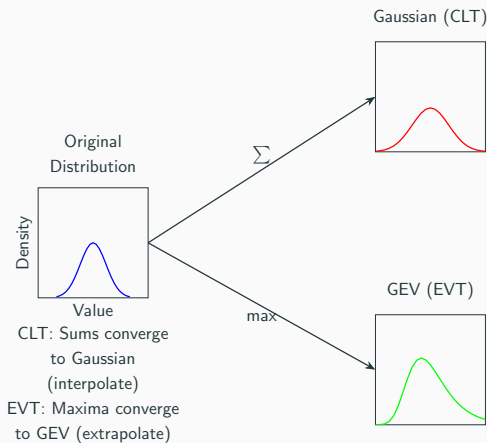
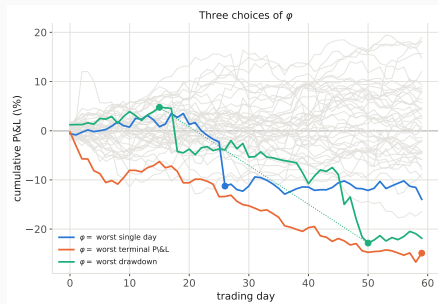


Figure 1: Extrapolation versus interpolation.

What happens when the rare event is not well specified by max?

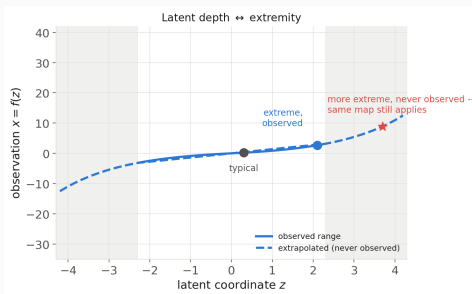
Simulate many scenarios for a book's P&L over time. Which path is “the” tail event depends on which risk metric φ you ask about:



Worst single day is a coordinatewise min, others require pathwise statistics. How do we generalize max such that we have semantically meaningful extrapolation from EVT?

Two design conditions

1. Extremes in the latent domain correspond to extremes in the observation domain.
2. Extrapolation in the latent domain enables extrapolation in the observation domain.



From max to a generalized operator φ

max is idempotent, commutative, and associative, but only sees magnitude in the given coordinates. Two cointegrated stocks can stay in range while their *spread* (a transformation of the two) reaches a rare extreme, invisible to max. Generalize it:

$$\varphi(X_i, X_i) = X_i, \quad \varphi(X_i, X_j) = \varphi(X_j, X_i), \quad \varphi(\varphi(X_i, X_j), X_k) = \varphi(X_i, \varphi(X_j, X_k)).$$

$$\begin{array}{ccc} \{X_1, \dots, X_n\} & \xrightarrow{\varphi} & X^{(n)} = \varphi(\{X_i\}) \\ \downarrow g \text{ (encoder)} & & \downarrow g \\ \{Z_1, \dots, Z_n\} & \xrightarrow{\max} & Z^{(n)} = \max(\{Z_i\}) \end{array}$$

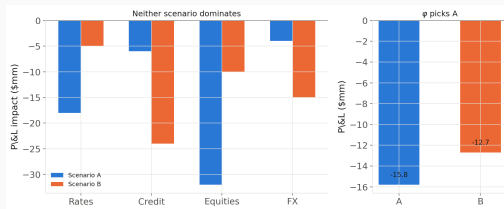
Definition (φ -stable). P is φ -stable if an invertible g exists with $g(\varphi(\{X_i\})) = \max(\{g(X_i)\})$ and $g(X)$ max-stable.

φ in finance: worst-loss scenario selection

φ picks, from a set of scenarios, whichever is worst for a given portfolio. Not a coordinate-wise max: neither scenario below is worse on every desk, yet φ still picks a definite answer.

- **Idempotent:** ranking a scenario against itself changes nothing.
- **Commutative:** ranking order doesn't matter.
- **Associative:** rank any subset of scenarios, then rank the winners.

φ -stability provides a latent space where this ranking becomes a coordinate-wise max, so EVT's extrapolation tools apply there directly.



Architecture: the de Haan spectral representation

$$z = \max_{i \leq n} (g(x_i) - \log A_i), \quad A_i = \sum_{j \leq i} \text{Exp}(1)$$

A_i are Poisson arrival times; this is the classical max-stable construction (de Haan), implemented as a stochastic max-pool over encoded samples.

Max-Stable Prior

This is amortized inference with a max-stable prior: the max-pool above is a reparameterization trick for sampling a max-stable z , similar to location-scale transform plays for a Gaussian prior. Training instead matches the decoded block-extreme to the true one directly (MSE); max-stability of z comes from the *construction* of the architecture.

The governing theorem: Fisher–Tippett–Gnedenko

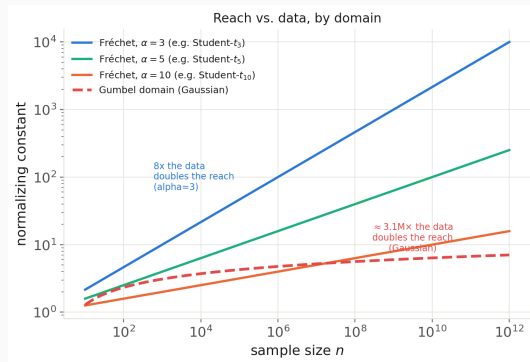
Remark (Condition for Extrapolation).

If the true model $X = f(z)$ satisfies

$$f(a_n z + b_n) \stackrel{d}{=} \varphi(f(z_1), \dots, f(z_n)),$$

the model extrapolates according to φ .

Every stable tail falls into one of three domains of attraction, and the normalizing constants (a_n, b_n) are set by which domain applies.



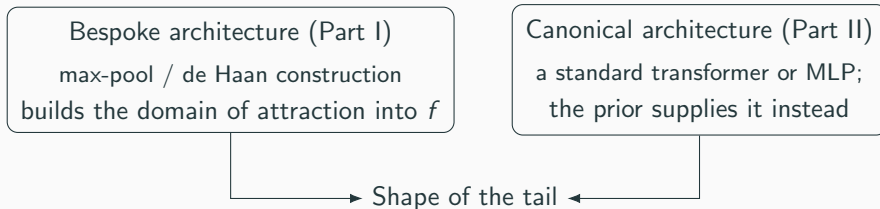
Recap

- We have a new way of defining what "maximum" makes sense for our data (φ)
- We have a way of using max stability to define this new type of stability for our data
- This gives us a new direction for how we can incorporate tail data

Part II: Shaping the Pretraining Distribution

From structural tails to pretraining tails

We now consider tails in the pretraining distribution:



How do we use these ideas with standard architectures (e.g. transformers)?

Context : $(x_1, y_1), \dots, (x_n, y_n)$ Query : $(x_{n+1}, ?)$

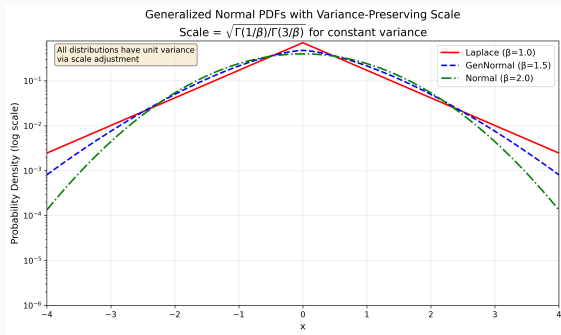
A transformer trained to predict the next observation under a simulator prior approximates that prior's posterior predictive distribution.

If we view in-context learning as amortized Bayesian inference, then how fast the posterior contracts is governed by the prior. How should we choose the prior for numerical tasks that may need extrapolation?

Concrete applications

- **Market making:** given context of current order book snapshot, what is the optimal controller for a given risk tolerance?
- **Execution:** optimal-stopping and trading-rate control has to generalize across contracts with different risk-aversion and volatility parameters, not just the ones seen in a backtest.
- **Fast Monte Carlo/PDE solving:** Applies to pricing across a family of dynamics and payoffs given current volatility surface representation, forward, etc.

The prior families



Controlling for location and scale, only tail shape varies: Student- $t(\nu)$ has polynomial tails, $\pi(\theta) \propto |\theta|^{-(\nu+1)}$; Generalized Normal(β) has exponential tails, $\pi(\theta) \propto \exp(-|\theta/\alpha|^\beta)$. $\nu = \infty$ and $\beta = 2$ both recover the Gaussian.

Theorem 1: generalization

$\widehat{\text{gen}}$ is the generalization gap: expected error on a freshly sampled task, minus error on the tasks already seen during pretraining.

$$\widehat{\text{gen}} \lesssim \frac{L_n \sqrt{n}}{\sqrt{M}} \left(1 + (\text{temporal dependence terms}) \right)$$

- L_n : model Lipschitz constant over an n -step context
- M : number of pretraining tasks
- the dependence terms are new: most prior ICL generalization work assumes i.i.d. context points, whereas financial sequences are temporally dependent by construction, and dependence increases the number of tasks needed.

$\Rightarrow$ Heavier tails and stronger dependence both increase the number of pretraining tasks needed for reliable generalization.

Theorem 2: task selection

$$\mathbb{E}[D_\alpha(\theta, \theta^*)] \leq \frac{1 + \alpha}{(1 - \alpha)n} \log \frac{1}{\pi(\theta^*)} + O\left(\frac{\log n}{n}\right)$$

The cost of identifying a task is the number of bits it costs under the prior.

$\implies$ the posterior over the task contracts at rate $\log(1/\pi(\theta^*))/n$. Heavier tails place more mass far from the bulk which accelerates identification of OOD tasks.

Theorem 1

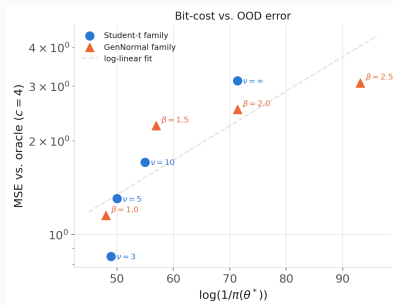
Heavier tails increase the number of tasks needed for generalization.

Theorem 2

Heavier tails reduce the cost of identifying a task far from the bulk.

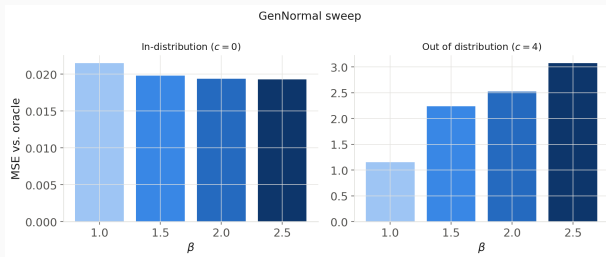
These point in opposite directions, implying an interior optimum in tail index.

Quantitative example



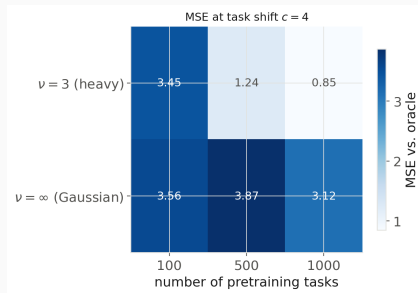
The Theorem 2 bit-cost, computed in closed form with no Monte Carlo, predicts the measured out-of-distribution gap across all 8 priors tested.

Generalized Normal sweep: a monotone reversal



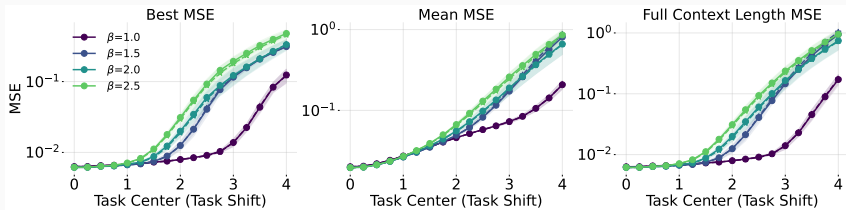
One parameter (β), monotone in both directions: lighter tail is better in-distribution, heavier tail is better out-of-distribution, with a crossover between $c = 1$ and $c = 2$.

Tails and diversity interact



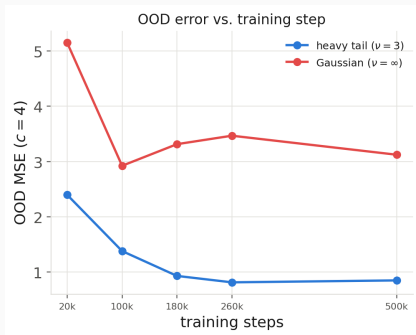
At $c = 4$: $\nu = 3$ improves $3.45 \rightarrow 1.24 \rightarrow 0.85$ as tasks go $100 \rightarrow 500 \rightarrow 1000$ ($4.1\times$); $\nu = \infty$ is flat (3.56, 3.87, 3.12). Task diversity pays off only if the prior has tails.

Example: OU prediction with shifts



The Gaussian-pretrained model's error rises with context length at $c = 4$: it reverts toward its prior rather than updating on evidence.

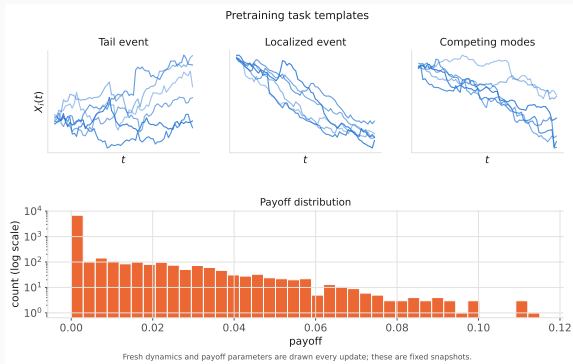
More training degrades out-of-distribution accuracy



Validation loss falls while out-of-sample performance under distribution shift falls with it.

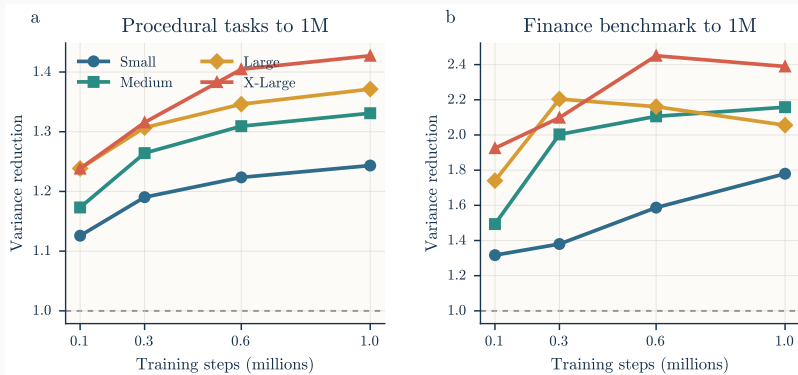
Part III: ICL for Monte Carlo

How this connects: what pretraining looks like here



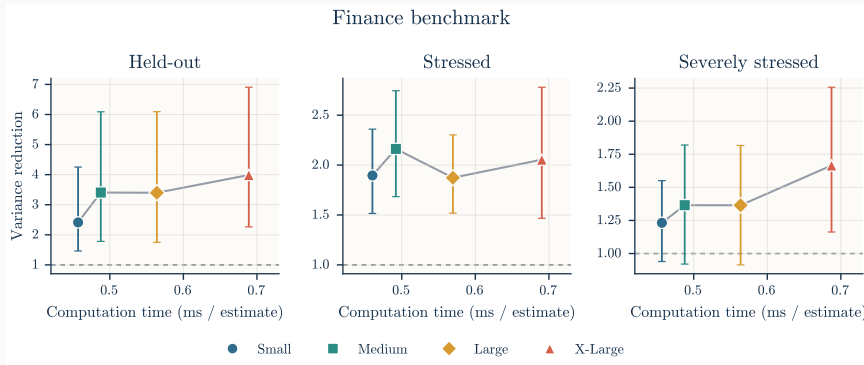
The pretraining distribution is now a distribution over dynamics and payoffs that are chosen to have a diverse set of outcomes. Payoffs are heavy-tailed by construction.

Pretraining scaling



Every capacity improves with more diverse pretraining, even on held out vocabularies and dynamics.

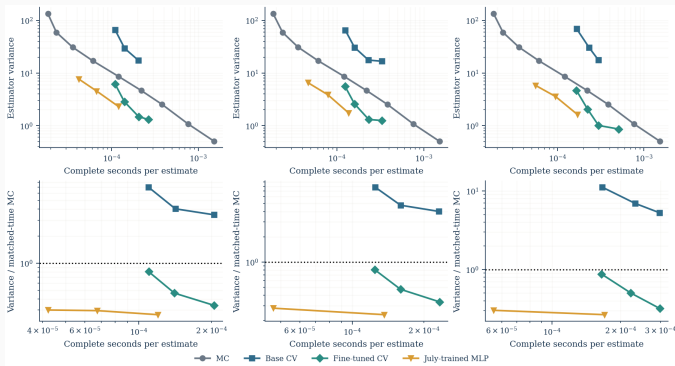
Cost vs. quality



Variance reduction from a pretrained estimator over plain Monte Carlo, across model capacity (Small→X-Large) and increasing distribution shift, vs. compute cost per estimate.

Post-training?

We'll use the base model to fine-tune with a bit of limit order book data to see how well it adapts to a totally new regime for optimal inventory holding.



Part IV: Synthesis

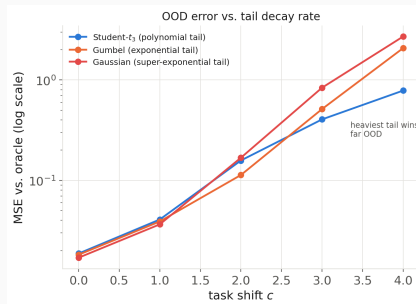
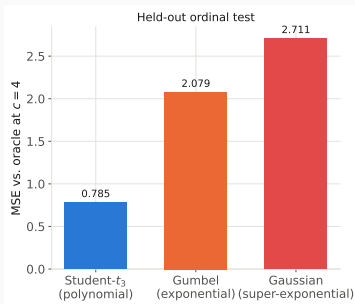
Missing Experiments

	buys new “magnitude”	buys new regime
in the architecture	Part I (φ -stable)	–
in the prior	–	Part II (heavy tails)

Using max-stable pretraining priors for ICL naturally follows from Parts I and II.

Testing the prediction

A Gumbel prior (exponential tail, a genuinely different family) pretrained against Student- t_3 and Gaussian on the OU task, matched to the same 60k-step budget:



This is as expected when checking the pretraining distribution. Now is there a way to exploit the max-stability?

Questions we want to answer:

- Does including φ -stability within the architecture itself improve in-context learning for OOD shifts?
- Does regularizing the evolution of the latent states have an effect similar to changing the pretraining distribution?

Practitioner checklist

1. Identify (or learn) the right φ for the problem you care about. It is often a transformation (e.g. worst running drawdown along a path, not the worst single day), not a coordinatewise max or min.
2. Budget pretraining diversity to match: heavy tails require more tasks before they help.

- The tails of the pretraining distribution have significant impact on out of distribution performance
- We can control the tail either architecturally (Part I) or through sampling (Part II), and each leads to different performance tradeoffs.
- Together, these results give practitioners two concrete recipes for building foundation models for numerical data that extrapolate.

Thank you!

Thank you!

Based on:

- Learning max-stable representations that extrapolate (UAI 2026)
- How does the pretraining distribution shape in-context learning? A fundamental tradeoff (ICML 2026)