FLOWTSFM: TURNING ENCODER DEPTH INTO QUANTILE TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Encoder-based time series foundation models (TSFMs) stack independently parameterized Transformer layers whose intermediate states carry no predictive meaning. We introduce FlowTSFM, which recasts encoder depth as *transport over predictive quantiles*. A single weight-shared Transformer block is integrated as a depth-conditioned residual update, and a shared decoder reads out a full quantile forecast at every step. A quantile-flow objective, inspired by flow matching, combines pinball supervision with a path term that moves the decoded quantiles along a straight line from an analytic Gaussian source to the terminal forecast. We prove that this position-matching term has the same zero-loss trajectories as regressing decoded velocities onto the constant flow-matching velocity. With 38.8M parameters, FlowTSFM is competitive with state-of-the-art TSFMs on GIFT-Eval and TIME and comes within 3–5% of the 119.5M-parameter Chronos-2. Its decoded updates consistently point toward the final forecast (CosMean 0.92, versus 0.35 for layer-wise probes of Chronos-2), and in our setup training without the path term diverged. A complete 5×5 training/inference depth study shows that fixed-depth training specializes the recurrence to its budget. Training with randomized depth removes this specialization: a single checkpoint improves with inference depth and matches or exceeds depth-specialized models across budgets.

1 INTRODUCTION

Time series foundation models (TSFMs) have made zero-shot probabilistic forecasting practical. A single pretrained model such as Chronos-2, TimesFM, Moirai, or TiRex now forecasts series it has never seen, across domains and sampling frequencies (Ansari et al., 2025; Das et al., 2024; Liu et al., 2025a; Auer et al., 2025). What these models share is less visible: they treat depth the way language models do, as a fixed stack of distinct layers of which only the last produces a forecast.

This choice has costs that show up in deployment. The interior of the network is opaque. If we decode Chronos-2 layer by layer, its intermediate predictions barely point toward its final answer, with an average cosine of 0.35 on GIFT-Eval (Section 5.2). Compute is fixed once training ends, so a cheaper forecast requires training and serving a second model. And every additional layer adds parameters.

Generative modeling suggests another way to think about depth. In flow matching (Lipman et al., 2023; Liu et al., 2023), computation transports a simple source distribution toward a target along a prescribed path, and each intermediate point has a meaning. Forecasting has an obvious source: a Gaussian centered on the context mean, with the context's standard deviation. We ask whether the depth of a forecasting encoder can be organized in the same way, so that the model starts from this naive forecast and refines it, step by step, into its final prediction.

FlowTSFM does this with three ingredients (Figure 2). A single weight-shared Transformer block is applied K times as a residual update of size $\frac{1}{K}$, conditioned on the current depth and on K itself (Dehghani et al., 2019; Chen et al., 2018). A shared decoder reads a full set of quantiles from every intermediate state. A quantile-flow objective trains these readouts to follow a straight line from the Gaussian quantiles to the final forecast, while pinball losses keep each step accurate. Nothing is sampled: the model moves quantiles, and we show that its path loss vanishes on exactly the trajectories where the flow-matching velocity loss vanishes (Proposition 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

With 38.8M parameters, FlowTSFM outperforms Sundial and TimesFM 2.0 on GIFT-Eval and TIME (Aksu et al., 2024; Qiao et al., 2026) and comes within 3–5% of Chronos-2, a model three times its size. Its interior behaves as intended: decoded updates point toward the final forecast with an average cosine of 0.92, and every attempt to train without the path term diverged. Depth, however, turns out to be less flexible than the ODE view suggests. In a complete 5×5 study, a model trained at one depth performs best only at that depth, and shortening a 12-step model to 4 steps degrades CRPS by up to 89%. Training with randomized depth removes this limitation: one checkpoint improves as depth grows and stays close to separately trained models at every budget (Figure 1).

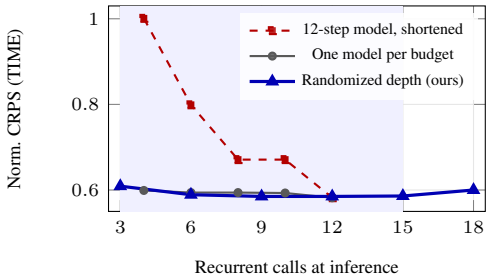


Figure 1: A single randomized-depth checkpoint (blue; shaded: training range) serves every budget. A fixed-depth model cannot be shortened (red).

Our contributions are:

- FlowTSFM, to our knowledge the first TSFM whose encoder depth is a weight-shared recurrence that yields a probabilistic forecast at every step, trained with a quantile-flow objective whose connection to flow matching we make precise.
- A complete training-depth $\times$ inference-depth study of recurrent forecasting, which shows that fixed-depth models specialize to their budget and that randomized-depth training turns depth into a test-time compute knob.
- Evidence that supervising decoded forecasts along a path stabilizes a shared recurrence that otherwise diverges, complementing latent-space stabilizers for looped language models (Prairie et al., 2026; Yang et al., 2026).

2 Related Work

FlowTSFM sits at the intersection of five lines of work: time series foundation models, weight-shared recurrent depth, adaptive and anytime computation, continuous-depth dynamics, and flow-based transport. Table 1 summarizes how it relates to the closest representatives. Appendix A gives an extended discussion.

Time series foundation models Pretrained forecasters now span encoder, decoder, and encoder–decoder Transformers: Chronos and Chronos-2 (Ansari et al., 2024; 2025), TimesFM (Das et al., 2024), the Moirai family (Woo et al., 2024; Liu et al., 2024a; 2025a), Timer and Timer-XL (Liu et al., 2024c; 2025b), Time-MoE (Shi et al., 2025), MOMENT (Goswami et al., 2024), Lag-Llama (Rasul et al., 2023), TimeGPT (Garza & Mergenthaler-Canseco, 2023), Toto (Cohen et al., 2025), and compact models such as TTM (Ekambaram et al., 2024). Other approaches repurpose language models (Gruver et al., 2023; Zhou et al., 2023; Jin et al., 2024) or use recurrence *across time* (Auer et al., 2025; Podest et al., 2026). Recent models increasingly predict quantiles directly with a pinball loss (Ansari et al., 2025; Liu et al., 2025a; Auer et al., 2025); Sundial and TSFlow instead generate samples with flow matching (Liu et al., 2025c; Kollovieh et al., 2025). Across all of these models, depth is a fixed stack of distinct layers and only the last layer produces a forecast. FlowTSFM changes this: it recurs across *depth*, and every step yields a forecast.

Weight-shared and looped depth Universal Transformers apply one block recurrently (Dehghani et al., 2019), and ALBERT showed that cross-layer sharing preserves much of the accuracy at a fraction of the parameters (Lan et al., 2020; Takase & Kiyono, 2023). Looped Transformers are expressive (Giannou et al., 2023; Yang et al., 2024; Saunshi et al., 2025; Xu & Sato, 2025), generalize to longer inputs (Fan et al., 2025), and scale test-time compute in language models (Geiping et al., 2025; Zhu et al., 2025; Bae et al., 2025a;b; Csordás et al., 2024; Schwethelm et al., 2026) and small recursive reasoners (Wang et al., 2025; Jolicoeur-Martineau, 2025). Shared recurrence has begun to reach other modalities: tabular foundation models (Rezaei Balef et al., 2026), graphs (Sun et al., 2026), and time series *classification* with looped state-space models (Farsang et al., 2026). A recurring obstacle is *instability* as the loop count grows. It is addressed through residual scaling

Table 1: **Positioning.** ✓: property holds; ○: partially or in a different space; –: absent. "Forecast every step": every intermediate state is decoded into a prediction. "Path supervision": intermediate outputs are trained toward a prescribed trajectory.

Method	Shared depth	Forecast/output every step	Path supervision	Quantile forecast	Variable test-time depth
Chronos-2, Moirai 2.0, TimesFM	–	–	–	✓	–
TiRex (Auer et al., 2025)	○ (time)	–	–	✓	–
Sundial, TSFlow	–	–	○ (samples)	–	○ (ODE steps)
Universal Transf., recurrent-depth LMs	✓	–	–	–	✓
Early exit (BranchyNet, CALM)	–	✓	–	–	✓
Elastic looped (LoopFormer, ELT)	✓	✓	○ (distill.)	–	✓
Looped SSM (Farsang et al., 2026)	✓	–	–	–	○
FlowTSFM (ours)	✓	✓	✓	✓	✓

(Wang et al., 2026), spectral constraints (Prairie et al., 2026), and Jacobian regularization combined with random loop sampling (Yang et al., 2026). To our knowledge, FlowTSFM is the first weight-shared recurrent-depth architecture for probabilistic forecasting. It stabilizes the recurrence through *output-space* supervision rather than constraints on the latent dynamics (Section 5.2).

Adaptive, anytime, and multi-budget computation Adaptive computation time and PonderNet learn when to halt (Graves, 2016; Banino et al., 2021). Early-exit networks attach predictors to intermediate layers (Teerapittayanon et al., 2016; Huang et al., 2018; Kaya et al., 2019; Elbayad et al., 2020; Xin et al., 2020; Schuster et al., 2022; Elhoushi et al., 2024), typically trained with deep supervision (Szegedy et al., 2015; Lee et al., 2015). A single set of weights can serve several budgets when depth or width is randomized during training (Huang et al., 2016; Fan et al., 2020; Yu et al., 2019; Cai et al., 2020; Kusupati et al., 2022). Looped models achieve the same by sampling loop counts (Geiping et al., 2025; Kuo et al., 2026) or conditioning on the budget, with intermediate outputs distilled toward the final one (Jeddi et al., 2026; Goyal et al., 2026). Iterative-refinement models with a shared update operator likewise supervise every iteration (Carreira et al., 2016; Teed & Deng, 2020). Our randomized-depth training follows this line. What FlowTSFM adds is a *prescribed geometry* for the intermediate outputs: each exit is a full quantile forecast that must lie on a straight path from an analytic Gaussian source to the terminal forecast. It is not merely pulled toward the target or toward the final exit.

Continuous-depth dynamics Residual networks can be read as discretized dynamical systems (E, 2017; Haber & Ruthotto, 2017; Lu et al., 2018; Chen et al., 2018). This view explains why unconstrained shared updates may explode (Pascanu et al., 2013; Haber & Ruthotto, 2017) and motivates regularizers that favor simple, nearly straight trajectories (Finlay et al., 2020; Kelly et al., 2020). For time series, this perspective has mostly been applied along the time axis (Rubanova et al., 2019; Kidger et al., 2020). FlowTSFM applies it along depth: its Euler update is budget-conditioned in the style of FiLM/adaLN (Perez et al., 2018; Peebles & Xie, 2023), and its path term is a trajectory regularizer imposed on decoded outputs rather than on the latent state.

Flow matching and transport Flow matching and stochastic interpolants regress velocity fields along prescribed probability paths (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Tong et al., 2024). Rectified flow favors straight paths (Liu et al., 2023), and shortcut, consistency, and mean-flow models condition on the step size or enforce consistency across step counts to enable few-step generation (Frans et al., 2025; Song et al., 2023; Geng et al., 2025). FlowTSFM borrows three ingredients: a straight interpolation path, an analytic source, and step-size conditioning. It applies them in a different space, to the *marginal quantiles* of a deterministic forecaster along encoder depth rather than to samples along generation time. Proposition 1 makes this link precise. Quantile regression (Koenker & Bassett, 1978), multi-quantile forecasters (Wen et al., 2017; Gasthaus et al., 2019), and CRPS (Gneiting & Raftery, 2007) provide the probabilistic backbone.

3 FLOWTSFM

FlowTSFM treats encoder depth as a path. The path starts at a naive Gaussian forecast and ends at the model's final forecast, and every step along the way is itself a forecast. This section describes the coordinates in which the path lives, the recurrent encoder that follows it, the loss that shapes it, and how the number of steps can change at inference.

3.1 COORDINATES AND THE GAUSSIAN SOURCE

Let x_{raw} be the context and m its observation mask. As in Chronos-2 (Ansari et al., 2025), we normalize with the mean μ and standard deviation σ of the observed values (Kim et al., 2022) and work with $x = \operatorname{arcsinh}(\frac{(x_{\text{raw}}-\mu)}{\sigma})$. The $\operatorname{arcsinh}$ is close to the identity for small values and grows like a logarithm for large ones, which keeps spikes from dominating. Targets y_s use the same map, and forecasts are returned to the original scale with $\mu + \sigma \sinh(\cdot)$.

The model predicts Q quantile levels $\tau_1 < \cdots < \tau_Q$ at T output positions, of which the loss scores the subset $\mathcal{M}$ (patching details in Appendix E). As the starting point of the path we use the *analytic source*

$$r_j = \operatorname{arcsinh}\big(\Phi^{-1}(\tau_j)\big), \qquad j = 1, \dots, Q,$$

repeated at every position, where Φ is the standard normal CDF. Mapped back to the original scale, r_j becomes $\mu + \sigma\Phi^{-1}(\tau_j)$, the τ_j-quantile of $\mathcal{N}(\mu, \sigma^2)$. The source is therefore the forecast one would make knowing only the level and spread of the context, and FlowTSFM refines it from there.

3.2 ONE BLOCK, APPLIED K TIMES

A standard encoder applies K different layers, $h^{k+1} = B_{\theta_k}(h^k)$. FlowTSFM applies the same block K times, as the steps of a discretized ODE (Chen et al., 2018; Dehghani et al., 2019):

$$h^{k+1} = h^k + \Delta t\, F_\theta\big(h^k,\, e_t(t_k),\, e_{\text{depth}}(K)\big), \qquad t_k = \tfrac{k}{K}, \quad \Delta t = \tfrac{1}{K}, \quad k = 0, \dots, K-1. \tag{1}$$

Depth t_k always runs from 0 to 1, whatever K is. The block is told where it is on the path through $e_t(t_k)$, and how many steps it has in total through $e_{\text{depth}}(K)$, both injected by adaptive layer normalization (Perez et al., 2018; Peebles & Xie, 2023; Frans et al., 2025). The budget signal is needed because with 4 steps each update must do three times as much work as with 12. It also means that changing K changes the dynamics themselves, not only their discretization (Section 5.3).

The initial state h^0 comes from a single unshared Transformer layer, run once on the input patches, which carry positional embeddings (Nie et al., 2023), a learned projection of r, and four register tokens (Darcet et al., 2024). A shared decoder maps any state to quantiles, $q^k = D_\phi(h^k) \in \mathbb{R}^{T\times Q}$, so every step is an *exit* with a complete forecast (Figure 2; details in Appendix E). Because F_θ and D_ϕ are shared, the parameter count does not depend on K.

3.3 THE QUANTILE-FLOW OBJECTIVE

A shared decoder makes the intermediate forecasts readable, but nothing yet makes them sensible: q^3 could be arbitrary as long as q^K is good. The objective fixes this with two terms, one for accuracy and one for the route.

Accuracy is measured at each exit with the pinball loss (Koenker & Bassett, 1978),

$$\ell^k_{\text{pin}} = \frac{1}{|\mathcal{M}|Q} \sum_{s\in\mathcal{M}} \sum_{j=1}^{Q} \rho_{\tau_j}(y_s - q^k_{s,j}), \qquad \rho_\tau(u) = \max\{\tau u, (\tau-1)u\}, \tag{2}$$

whose minimizer is the true τ-quantile. For the route, we ask each exit to lie on the straight line from the source to the final forecast,

$$\pi^k = (1 - t_k)\, r + t_k\, \operatorname{sg}(q^K), \tag{3}$$

and penalize the squared distance to it,

$$\text{MSE}(A, B) = \frac{1}{|\mathcal{M}|Q} \sum_{s\in\mathcal{M}} \sum_{j=1}^{Q} (A_{s,j} - B_{s,j})^2. \tag{4}$$

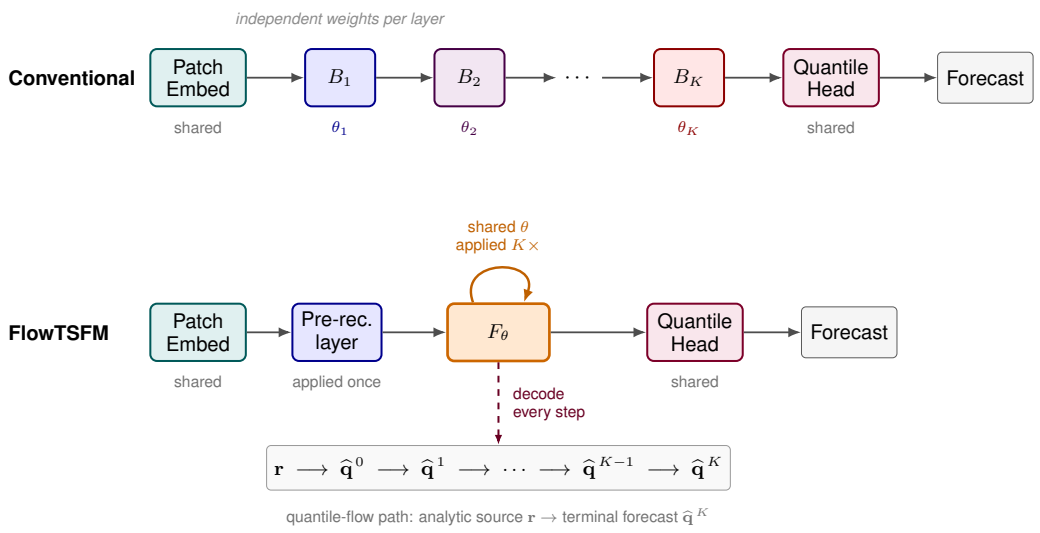


Figure 2: **Conventional versus recurrent encoder depth.** FlowTSFM replaces independent blocks with one shared block applied K times after a pre-recurrent layer. The same decoder reads every intermediate state, so depth becomes a sequence of forecasts from the Gaussian source r to the final forecast q^K.

Halfway through the recurrence, the forecast should be halfway between the Gaussian and the final answer. The stop-gradient sg keeps the endpoint fixed as a target, so the path term cannot be reduced by pulling the final forecast back toward the Gaussian.

To keep training cheap, each step scores only $m_{\text{exit}} = 3$ interior exits $\mathcal{E}' \subseteq \{1, \ldots, K-1\}$, drawn at random. The full loss is

$$\mathcal{L} = \underbrace{\ell^K_{\text{pin}}}_{\text{final forecast}} + \underbrace{\lambda_0 \,\text{MSE}(q^0, r)}_{\text{start at the source}} + \sum_{k \in \mathcal{E}'} \Big[\underbrace{\lambda_{\text{ds}}\, t_k\, \ell^k_{\text{pin}}}_{\text{accurate exits}} + \underbrace{\lambda_{\text{qf}}\, \text{MSE}(q^k, \pi^k)}_{\text{on the path}} \Big]. \tag{5}$$

The anchor on q^0 is soft, since the first readout may already reflect what the pre-recurrent layer has seen. The weight t_k asks less of early exits than of late ones, as in deep supervision (Lee et al., 2015).

Relation to flow matching Flow matching supervises velocities. Along a straight path, it regresses the velocity onto the constant displacement between the endpoints (Lipman et al., 2023). We supervise positions, because positions are what the decoder outputs and what a user reads. The two are closely linked. Write $v^k = \frac{(q^{k+1} - q^k)}{\Delta t}$ for the decoded velocity and $u = \text{sg}(q^K) - r$ for the reference velocity.

Proposition 1 (Quantile-path matching as integrated velocity regression)**.** *Fix $K \geq 2$ and decoded readouts $q^0, \ldots, q^K$. For every interior exit,*

$$\text{MSE}(q^k, \pi^k) = \text{MSE}\Big(q^0 - r + \Delta t \sum_{i=0}^{k-1} (v^i - u),\, 0\Big), \tag{6}$$

and, if $q^0 = r$, $\text{MSE}(q^k, \pi^k) = t_k^2\, \text{MSE}(\bar{v}^{\,k}, u)$ with $\bar{v}^{\,k} = \frac{1}{k}\sum_{i<k} v^i$. Moreover, with $\mathcal{R}_{\text{path}} = \text{MSE}(q^0, r) + \sum_{k=1}^{K-1} \text{MSE}(q^k, \pi^k)$ and $\mathcal{R}_{\text{vel}} = \sum_{k=0}^{K-1} \text{MSE}(v^k, u)$,

$$\tfrac{1}{4K^2}\mathcal{R}_{\text{vel}} \;\leq\; \mathcal{R}_{\text{path}} \;\leq\; \tfrac{K+1}{2K}\mathcal{R}_{\text{vel}}, \tag{7}$$

so the two losses vanish on exactly the same trajectories.

Eq. 6 says that the path loss at step k measures the velocity error accumulated so far. A step that overshoots can be corrected by the next one, which is more permissive than matching each velocity separately, yet both losses are zero only on the straight path. The proof is in Appendix B. The result is about loss values at a fixed depth, not about gradients or the minimizers of the full objective.

3.4 CHANGING THE NUMBER OF STEPS

Since the block is shared, the model can be run with any number of steps n. We set $t_k = \frac{k}{n}$, $\Delta t = \frac{1}{n}$, and condition on $e_{\text{depth}}(n)$. The rollout still covers the whole path, with fewer and longer steps or more and shorter ones. Early exiting is different: it keeps the training step size and stops partway. Only the final state is decoded, so a forecast costs $C(n) = C_{\text{input}} + n\,C_F + C_D$, where C_F is one block call.

3.5 RANDOMIZED-DEPTH TRAINING

Eq. 5 constrains one rollout at a time, so a model trained only with $K = 12$ steps has no reason to work well with 4. Following looped language models (Geiping et al., 2025; Kuo et al., 2026; Jeddi et al., 2026), we therefore vary the depth during training. At every step we draw K uniformly from $\{3, \dots, 15\}$ and apply Eq. 5 to a rollout of that length, with its own grid, step size, and conditioning $e_{\text{depth}}(K)$. A single set of weights then minimizes

$$\mathcal{J}_{\text{rd}} = \frac{1}{13} \sum_{K=3}^{15} \mathbb{E}\big[\mathcal{L}^{(K)} \mid K\big], \tag{8}$$

where the expectation is over training examples and exit subsets. The number of steps then becomes a setting chosen at inference, trading compute for accuracy without retraining.

4 EXPERIMENTAL SETUP

Models The base FlowTSFM uses a context of 8,192 steps, patches of 16, width 1,024, 16 heads, 4 registers, one pre-recurrent layer, 99 output quantiles, and $K = 12$ recurrent steps, for 38.8M parameters. It is trained for 100,000 AdamW steps with batch size 1,024 and gradient clipping at 1.0, on an equal mixture of domain-balanced GIFT-Eval-Pretrain and kernel-synthetic series (Aksu et al., 2024; Ansari et al., 2024). The depth study uses cheaper runs trained with batch size 128: five fixed-depth checkpoints with $K_{\text{train}} \in \{4, 6, 8, 10, 12\}$ and one randomized-depth checkpoint with $K \sim \text{Unif}\{3, \dots, 15\}$, all otherwise identical. Table 5 lists every hyperparameter.

Evaluation We report MASE and CRPS normalized by Seasonal Naive and aggregated by geometric mean, on GIFT-Eval (97 configurations over 55 datasets) and TIME (98 configurations) (Aksu et al., 2024; Qiao et al., 2026). Lower is better. For the depth study, S_{ij} denotes the score of the checkpoint trained with K_i steps and run with K_j, and

$$\delta_{ij} = 100\,\big(\frac{S_{ij}}{S_{ii}} - 1\big) \tag{9}$$

is the relative loss from running it at a budget other than its own.

5 RESULTS

We answer the three questions raised in the introduction in turn: whether FlowTSFM is competitive, whether its interior behaves as a transport, and whether its depth can be chosen at inference.

5.1 FLOWTSFM IS COMPETITIVE AT A THIRD OF THE SIZE

Table 2 compares FlowTSFM with leading TSFMs. With normalized MASE/CRPS of 0.729/0.500 on GIFT-Eval and 0.694/0.580 on TIME, it outperforms Sundial and TimesFM 2.0 on all four scores and Moirai 2.0 on three. Chronos-2, with three times as many parameters, is 3–5% more accurate. TiRex, of similar size, is 1–2% more accurate; its recurrence runs across time rather than depth, and the two ideas could be combined. The depth-study checkpoints below score slightly lower than the base model (0.743/0.506 on GIFT-Eval at 12 steps), as expected from their eight times smaller batch; every comparison within the depth study uses identically trained checkpoints.

Table 2: **Forecasting accuracy study.** Seasonal-Naive-normalized errors (lower is better) and reported parameter counts. The FlowTSFM row corresponds to the base accuracy evaluation; depth-sweep results are in Table 6.

	GIFT-Eval ↓		TIME ↓		
Model	MASE	CRPS	MASE	CRPS	Parameters
FlowTSFM (base)	0.729	0.500	0.694	0.580	38.8M
Sundial (base)	0.750	0.559	0.758	0.663	128.3M
Tirex	0.716	0.488	0.683	0.573	35.3M
Moirai-2	0.728	0.516	0.703	0.588	**11.4M**
TimesFM 2.0	0.758	0.550	0.718	0.620	498.8M
Chronos-2	**0.697**	**0.485**	**0.662**	**0.556**	119.5M

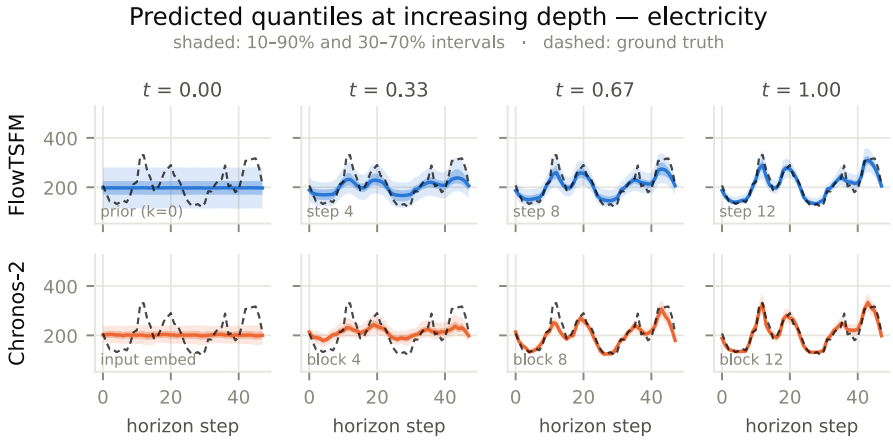


Figure 3: **Forecasts along depth on an Electricity-H series.** Solid: median; shading: 30–70% and 10–90% intervals; dashed: ground truth. FlowTSFM (top) starts from the Gaussian source and sharpens the forecast step by step. Chronos-2 (bottom), decoded after each block with its own head, stays uninformative until its final blocks.

5.2 The interior follows the prescribed transport

Figure 3 shows what the quantile-flow objective produces on a single series. FlowTSFM's first readout is the flat Gaussian band; daily peaks appear after a few steps, and the remaining steps sharpen the intervals. In contrast, decoding Chronos-2 block by block reveals uneven progress toward the terminal forecast across the prediction horizon: some horizon positions are already close to their final predictions at intermediate depths, while others undergo substantial corrections in later layers.

To measure this over a whole benchmark, we compute how well each update points toward the final forecast:

$$\mathrm{CosMean} = \frac{1}{K-1}\sum_{k=0}^{K-2} \cos\big(q^{k+1} - q^{k},\ q^{K} - q^{k}\big) \in [-1, 1]. \tag{10}$$

A value of 1 means that every step moves straight toward the final answer. On GIFT-Eval, using the nine quantile levels both models share, FlowTSFM reaches 0.919 averaged over datasets (0.933 over series), against 0.350 (0.317) for Chronos-2 (Appendix C). FlowTSFM is trained toward this geometry, so the gap is best read as evidence that the objective does what it is designed to do, and that a conventionally trained encoder does not organize its depth this way on its own.

The path also stabilizes training The path term turns out to be more than a way to organize the interior. Removing it ($\lambda_{\mathrm{qf}} = 0$), while keeping the terminal, deep-supervision, and source losses, made training diverge in both runs we attempted, despite gradient clipping. In a representative run, the gradient norm grows from 1.0 at step 1,700 to 1.5×10^{14} at step 2,900, and all losses are NaN

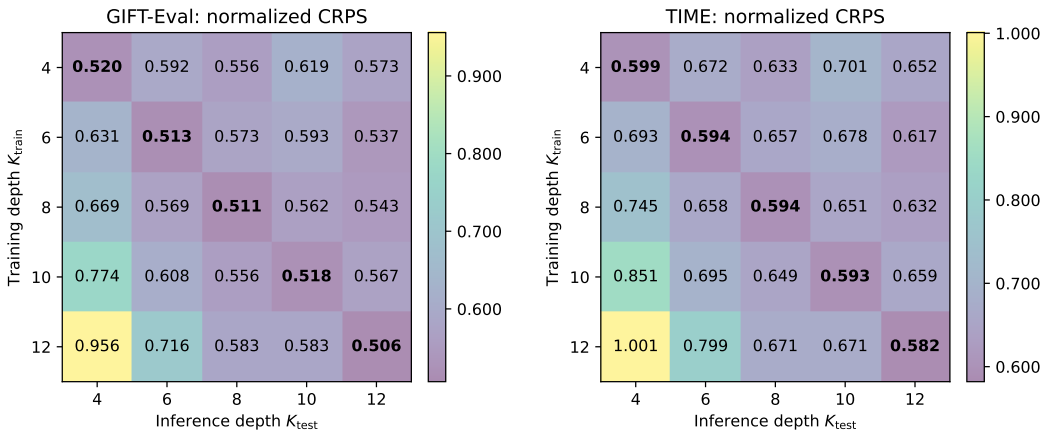


Figure 4: **Fixed-depth models only work at their training depth.** Normalized CRPS of five checkpoints (rows) run at five budgets (columns). Every row is best on the diagonal. MASE grids are in Appendix F.

Table 3: **Three ways to spend a compute budget** (normalized MASE / CRPS, lower is better). *Randomized*: one checkpoint, trained with $K \in \{3, \ldots, 15\}$. *Specialist*: a separate checkpoint trained at that exact budget. *Shortened*: the 12-step checkpoint run with fewer steps. –: no such checkpoint in the depth study.

	Recurrent steps	3	6	9	12	15	18*
GIFT-Eval	Randomized (one model)	.768/.530	**.748**/.517	.746/.516	.746/.516	.746/.517	.767/.529
	Specialist	–	.754/**.513**	–	**.743/.506**	–	–
	12-step, shortened	–	1.046/.716	–	**.743/.506**	–	–
TIME	Randomized (one model)	.735/.609	**.707/.589**	.701/.585	.701/.585	.702/.586	.716/.600
	Specialist	–	.711/.594	–	**.698/.582**	–	–
	12-step, shortened	–	1.002/.799	–	**.698/.582**	–	–

*Outside the training range.

from step 22,700 on (Appendix H). With the path term, the same configuration trains stably for 100,000 steps. Our interpretation is that deep supervision alone lets each exit move freely, and a shared block applied twelve times amplifies these moves; the path fixes where each exit should be and keeps the recurrence in check.

5.3 Randomized depth training enables test-time scaling

Fixed-depth training ties a model to its depth We first train five models with 4 to 12 steps and run each of them at all five budgets (Figure 4). Two findings stand out. On the diagonal, accuracy is remarkably flat: the 4-step model is only 2.3–3.0% worse than the 12-step model while using a third of the steps. Off the diagonal, accuracy drops sharply: every model is best at the depth it was trained with, the median penalty for a mismatched budget is 10–15%, and shortening the 12-step model to 4 steps raises the error by 72–89% (Appendix F). The reason is visible in Eq. 5: the loss constrains one rollout with one budget embedding, so nothing encourages rollouts of other lengths to reach the same forecast.

Randomized depth unties it Table 3 evaluates a single checkpoint trained with a random number of steps (Section 3.5). Its behavior is what a practitioner would want from a compute knob. Accuracy improves as steps are added, by about 3% on GIFT-Eval and 4–5% on TIME from 3 to 9 steps, and then levels off up to 15 steps, the largest depth seen in training. Where a specialist exists, the single randomized model is within 2% of it on every score and better on three of the eight; at 6 steps, it has 26–29% lower error than the shortened 12-step model. Beyond the training range, at 18 steps, accuracy degrades on both benchmarks. The budget embedding therefore interpolates between the

depths it was trained on but does not extrapolate beyond them, so the training range should cover the budgets expected at deployment. One model thus replaces a family of depth-specific models.

Sensitivity of the budget embedding To investigate the limits of test-time scaling, we probe $e_{\text{depth}}(K)$ for $K \in \{1, \ldots, 32\}$ in a fixed-$K = 12$ checkpoint and a randomized-depth checkpoint (Appendix H.1). At matched progress $t = 0.5$, changing the budget from 12 to 3 changes the summed conditioning $S(t, K) = e_t(t) + e_{\text{depth}}(K)$ by only 0.319% of its reference norm in the fixed-depth checkpoint, compared with 40.3% in the randomized-depth checkpoint. The fixed-depth budget embeddings also have pairwise cosine similarities above 0.9999, indicating little directional variation. Randomized-depth training is associated with a more budget-sensitive conditioning signal, but its response flattens beyond the training range: the embedding displacement from $K = 10$ to 15 is 9.1% of its starting norm, compared with 2.0% from 15 to 20 and 1.2% from 20 to 25. Thus, the explicit budget signal changes much less outside the sampled range even as the Euler step size $\Delta t = \frac{1}{K}$ continues to decrease. This saturation suggests a possible mechanism for the degradation at 18 steps in Table 3, rather than establishing its cause.

6 Conclusion

Time series foundation models have so far improved mostly by adding data, parameters, and layers. FlowTSFM shows that a different lever is available. When encoder depth is organized as transport from a Gaussian forecast to the final one, a single shared block is enough to reach accuracy close to models three times larger, the model's interior becomes a sequence of forecasts that can be read and checked, and the path that produces this interior is also what keeps the shared recurrence stable in training. Our depth study adds a practical lesson: a well-behaved trajectory does not by itself make a model robust to a new budget. Training at a fixed depth ties the model to that depth, whereas randomized-depth training gives one checkpoint that serves many budgets.

These results change what can be decided after training. With FlowTSFM, the cost of a forecast is chosen at deployment rather than fixed by the architecture, and the same weights can run with few steps on an edge device or with more steps in the cloud. Because every step yields a forecast, the model can also stop once successive forecasts no longer change, which opens the way to per-series adaptive computation: easy series would take few steps and hard ones would take more. The approach is also complementary to current designs. Recurrence across depth can be combined with recurrence across time (Auer et al., 2025), with multivariate attention (Ansari et al., 2025), or with larger training corpora, and supervising outputs along a path may help stabilize looped models beyond forecasting.

Limitations Baseline scores come from official leaderboards rather than retraining under a common data and compute budget. The CosMean comparison with Chronos-2 is a diagnostic, since Chronos-2 was not trained with intermediate objectives. Two controls would isolate our components more sharply: a loss-off model stabilized by other means, and an untied architecture with the same objective. We report single training runs, and calibration and quantile-crossing diagnostics are left to future work (Appendix I). Natural next steps are endpoint-consistency losses across budgets and learned, input-dependent stopping.

Reproducibility Statement

Appendix E specifies the full architecture, and Table 5 lists all training hyperparameters. Appendix B contains the complete proof of Proposition 1. Appendix D describes the evaluation protocol and the mapping from checkpoints to reported numbers, and Appendix F reports all 100 depth-study scores. We will release training and evaluation code, all checkpoints, and per-task forecasts upon acceptance.

Ethics Statement

Probabilistic forecasts inform consequential decisions in energy, healthcare, and operations, and miscalibrated uncertainty can lead to poor resource allocation. We therefore report both point (MASE) and probabilistic (CRPS) accuracy and discuss calibration as an open diagnostic. All training data are public benchmark corpora or synthetic series, and no personally identifiable information is used.

AI Use Statement

We used generative AI tools to polish the writing and to assist in writing code for the model and experiments. All AI-assisted code was reviewed and tested by the authors, who take full responsibility for the content of this work.

References

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024. URL https://arxiv.org/abs/2410.10393.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. *arXiv preprint arXiv:2505.23719*, 2025.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, 2025a.

Sangmin Bae et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025b.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 2019.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Nora Belrose et al. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020.

João Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1806.07366.

Victor Chernozhukov, Iván Fernández-Val, and Alfred Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010.

Ben Cohen et al. This time is different: An observability perspective on time series foundation models. *arXiv preprint arXiv:2505.14766*, 2025.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. In *Advances in Neural Information Processing Systems*, 2024.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International Conference on Machine Learning*, 2018.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K. Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020.

Mostafa Elhoushi et al. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Annual Meeting of the Association for Computational Linguistics*, 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, 2025.

Mónika Farsang, Ramin Hasani, Daniela Rus, and Radu Grosu. Looped SSMs: Depth-recurrence and input reshaping for time series classification. *arXiv preprint arXiv:2605.16048*, 2026.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M. Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *International Conference on Machine Learning*, 2020.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/559a0998fab1d19b80e7e43a5852401c-Abstract-Conference.html.

Azul Garza and Max Mergenthaler-Canseco. TimeGPT-1. *arXiv preprint arXiv:2310.03589*, 2023.

Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. arXiv preprint arXiv:2502.05171, 2025. URL https://arxiv.org/abs/2502.05171.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, 2023.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Sahil Goyal, Swayam Agrawal, Gautham Govind Anil, Prateek Jain, Sujoy Paul, and Aditya Kusupati. ELT: Elastic looped transformers for visual generation. *arXiv preprint arXiv:2604.09168*, 2026.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, 2023.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34 (1), 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

Jeddi et al. LoopFormer: Elastic-depth looped transformers for latent reasoning. *arXiv preprint arXiv:2602.11451*, 2026.

Ming Jin et al. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning*, 2019.

Jacob Kelly, Jesse Bettencourt, Matthew J. Johnson, and David Duvenaud. Learning differential equations that are easy to solve. In *Advances in Neural Information Processing Systems*, 2020.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. In *Advances in Neural Information Processing Systems*, 2023.

Marcel Kollovieh, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günnemann. Flow matching with Gaussian process priors for probabilistic time series forecasting. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.03024.

Hsun-Yu Kuo, El Mahdi Chayti, Patrik Reizinger, Wieland Brendel, and Martin Jaggi. Stabilizing extrapolation in looped transformers via learned stochastic stopping. *arXiv preprint arXiv:2606.29983*, 2026.

Aditya Kusupati et al. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, 2022.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 562–570. PMLR, 2015. URL https://proceedings.mlr.press/v38/lee15a.html.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Doyen Sahoo, Caiming Xiong, Silvio Savarese, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024b.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *International Conference on Machine Learning*, 2024c.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *International Conference on Learning Representations*, 2025b.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39295–39317. PMLR, 2025c. URL https://proceedings.mlr.press/v267/liu25be.html.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, 2018.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with Transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2211.14730.

nostalgebraist. Interpreting GPT: The logit lens. LessWrong blog post, 2020.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

Francesco Pappone, Donato Crisostomi, and Emanuele Rodolà. Two-scale latent dynamics for recurrent-depth transformers. *arXiv preprint arXiv:2509.23314*, 2025.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 2013.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Patrick Podest, Marco Pichler, Elias Bürger, Levente Zólyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex-2: Generalizing TiRex to multivariate data and streaming. *arXiv preprint arXiv:2607.01204*, 2026. URL https://arxiv.org/abs/2607.01204.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026.

Zhongzheng Qiao, Sheng Pan, Anni Wang, Viktoriya Zhukova, Yong Liu, Xudong Jiang, Qingsong Wen, Mingsheng Long, Ming Jin, and Chenghao Liu. It's TIME: Towards the next generation of time series forecasting benchmarks. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.12147.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, 2021a.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021b.

Kashif Rasul et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? understanding inference dynamics in tabular foundation models. In *International Conference on Machine Learning*, 2026.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2019.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022.

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. How much is one recurrence worth? iso-depth scaling laws for looped language models. *arXiv preprint arXiv:2604.21106*, 2026.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Junshu Sun, Wanxing Chang, Qingming Huang, and Shuhui Wang. Adaptive recurrent message passing for test time computing on graphs. In *International Conference on Machine Learning*, 2026.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. In *Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, 2023.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, 2021.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, 2020.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition*, 2016.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Guan Wang et al. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Shaowen Wang, Bingrui Li, Ge Zhang, Wenhao Huang, Shen Yan, and Jian Li. On the residual scaling of looped transformers: Stability and transferability. *arXiv preprint arXiv:2606.18524*, 2026.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 2024. URL `https://proceedings.mlr.press/v235/woo24a.html`.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. In *International Conference on Machine Learning*, 2025.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024.

Xiao-Wen Yang, Ziyu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing recurrent dynamics for test-time scalable latent reasoning in looped language models. In *International Conference on Machine Learning*, 2026.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2023.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 2022.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. BERT loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, 2020.

Rui-Jie Zhu et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Extended Related Work

This section expands Section 2 domain by domain and states, for each domain, what FlowTSFM borrows and where it departs.

A.1 Time series forecasting

Task-specific deep forecasters Before foundation models, deep forecasters were trained per dataset. Examples include autoregressive probabilistic RNNs such as DeepAR (Salinas et al., 2020); interpretable MLP stacks such as N-BEATS (Oreshkin et al., 2020) and TiDE (Das et al., 2023); attention-based models including TFT, Informer, Autoformer, and FEDformer (Lim et al., 2021; Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022); and patch- and variate-token Transformers (Nie et al., 2023; Liu et al., 2024b). TimesNet (Wu et al., 2023) and the strong linear baselines of Zeng et al. (2023) showed that architectural choices matter less than long assumed. FlowTSFM inherits patching (Nie et al., 2023) and reversible instance normalization (Kim et al., 2022).

Time series foundation models Large-scale pretraining for zero-shot forecasting (Liang et al., 2024) has produced several families. Tokenized language-model-style models include Chronos (Ansari et al., 2024). Decoder-only patch models include TimesFM, Timer, Timer-XL, Time-MoE, Moirai 2.0, and Toto (Das et al., 2024; Liu et al., 2024c; 2025b; Shi et al., 2025; Liu et al., 2025a; Cohen et al., 2025). Masked encoders include Moirai and Moirai-MoE (Woo et al., 2024; Liu et al., 2024a), and MOMENT is a multi-task encoder (Goswami et al., 2024). Chronos-2 (Ansari et al., 2025) is an encoder with group attention for multivariate and covariate-aware forecasting. Probabilistic decoder-only forecasters include Lag-Llama (Rasul et al., 2023) and the commercial TimeGPT (Garza & Mergenthaler-Canseco, 2023). TTM (Ekambaram et al., 2024) shows that compact MLP-mixer models can be competitive. TiRex and TiRex-2 (Auer et al., 2025; Podest et al., 2026) use xLSTM recurrence across time steps and are among the strongest compact models. LLM-reprogramming approaches (Gruver et al., 2023; Zhou et al., 2023; Jin et al., 2024) adapt text models to forecasting. Evaluation has consolidated around GIFT-Eval (Aksu et al., 2024) and, more recently, TIME (Qiao et al., 2026). All of these models give intermediate layers no predictive role. FlowTSFM is complementary: its recurrent-depth encoder and quantile-flow objective could in principle be combined with multivariate group attention or recurrence across time.

Probabilistic and quantile forecasting Quantile regression (Koenker & Bassett, 1978) with the pinball loss underlies multi-horizon quantile forecasters (Wen et al., 2017) and spline quantile-function models (Gasthaus et al., 2019). Quantile-based distributional modeling also appears in reinforcement learning (Dabney et al., 2018). Proper scoring rules, in particular CRPS (Gneiting & Raftery, 2007), are the standard evaluation. Non-crossing quantiles are a classical concern (Chernozhukov et al., 2010); because our path starts from a monotone Gaussian source, crossing rates along depth are a natural diagnostic for future work (Appendix I). Generative alternatives model the full predictive distribution with normalizing flows (Rasul et al., 2021b), diffusion (Rasul et al., 2021a; Tashiro et al., 2021; Kollovieh et al., 2023), or flow matching (Kollovieh et al., 2025; Liu et al., 2025c). FlowTSFM stays in the quantile paradigm but borrows the *transport* view from these generative models.

A.2 Weight-shared, looped, and recurrent-depth networks

Foundations and expressivity Universal Transformers (Dehghani et al., 2019) introduced depth recurrence with adaptive halting. ALBERT (Lan et al., 2020) and later studies of parameter sharing (Takase & Kiyono, 2023) showed that tied layers remain effective. Looped Transformers can emulate programmable computers (Giannou et al., 2023), learn iterative algorithms in context (Yang et al., 2024), simulate chain-of-thought with latent iterations (Saunshi et al., 2025), and generalize to longer inputs (Fan et al., 2025). Timestep-modulated looped Transformers (Xu & Sato, 2025) condition each loop on a step encoding, analogous to our $e_t(t_k)$. Deep equilibrium models push weight sharing to the infinite-depth fixed point (Bai et al., 2019).

Scaling test-time compute Recurrent-depth language models scale reasoning by unrolling more loops at inference (Geiping et al., 2025; Zhu et al., 2025; Snell et al., 2024). Parameter-efficient variants relax strict tying (Bae et al., 2025a), route tokens to different recursion depths (Bae et al.,

2025b), or combine recurrence with mixtures of experts (Csordás et al., 2024). Iso-depth scaling laws quantify how much a recurrence is worth relative to unique layers (Schwethelm et al., 2026). Small recursive reasoners (Wang et al., 2025; Jolicoeur-Martineau, 2025) and equilibrium reasoners (Huang et al., 2026) show that tiny shared networks can solve hard reasoning tasks through iteration. Mechanistic analyses find structured latent dynamics across loops (Pappone et al., 2025; Blayney et al., 2026).

Beyond language Looped architectures have recently been applied to visual generation (Goyal et al., 2026), graphs with step-conditioned test-time compute (Sun et al., 2026), tabular foundation models (Rezaei Balef et al., 2026), and time series classification with looped state-space models (Farsang et al., 2026). To our knowledge, FlowTSFM is the first recurrent-depth model for *probabilistic forecasting* and the first to use it inside a pretrained time series foundation model.

Stability of looped models Stable training of looped models is an active concern. Proposed remedies include residual scaling proportional to the inverse loop count (Wang et al., 2026), which matches our step size $\Delta t = 1/K$; spectral-norm constraints on the injection (Prairie et al., 2026); and Jacobian spectral-radius regularization combined with random loop sampling (Yang et al., 2026). These methods constrain the latent dynamics. Our ablation (Appendix H) suggests a complementary mechanism: supervising the decoded outputs along a prescribed path is enough to stabilize the recurrence in our setting.

A.3 Adaptive, anytime, and multi-budget computation

Halting and early exit Adaptive computation time (Graves, 2016) and PonderNet (Banino et al., 2021) learn input-dependent halting. Early-exit networks attach classifiers to intermediate layers (Teerapittayanon et al., 2016; Huang et al., 2018; Kaya et al., 2019) and were extended to Transformers (Elbayad et al., 2020; Xin et al., 2020; Zhou et al., 2020; Schuster et al., 2022; Elhoushi et al., 2024). Intermediate heads are usually trained with deep supervision (Szegedy et al., 2015; Lee et al., 2015). Our deep-supervision term follows this practice, and our path term adds a geometric constraint that early-exit objectives lack.

Training one model for many budgets Stochastic depth (Huang et al., 2016) and LayerDrop (Fan et al., 2020) randomize depth during training so that layers can be dropped at inference. Slimmable, once-for-all, and Matryoshka models (Yu et al., 2019; Cai et al., 2020; Kusupati et al., 2022) train a single set of weights for many widths or sizes. Among looped models, recurrent-depth language models sample loop counts during training (Geiping et al., 2025), stochastic loop counts with learned stopping reduce variance across inference budgets (Kuo et al., 2026), and elastic-depth looped Transformers condition on the budget and distill intermediate outputs toward the final one (Jeddi et al., 2026; Goyal et al., 2026). Our randomized-depth training (Section 3.5) belongs to this family. The difference is the target of the intermediate supervision: a straight path from an analytic source, applied to marginal quantiles.

Iterative refinement Models that repeatedly apply a shared update and supervise every iterate appear in pose estimation (Carreira et al., 2016) and optical flow (Teed & Deng, 2020). RAFT in particular weights later iterates more heavily, as our $\lambda_{\mathrm{ds}} t_k$ does. FlowTSFM brings this principle to probabilistic forecasting and adds the analytic source and the straight-path target.

Reading intermediate layers Linear probes (Alain & Bengio, 2016), the logit lens (nostalgebraist, 2020), and the tuned lens (Belrose et al., 2023) decode predictions from the intermediate layers of conventional networks. Our CosMean comparison with Chronos-2 is a logit-lens-style probe. The contrast it reveals, uninformative intermediate predictions in a conventional encoder versus steady progress in FlowTSFM, reflects what an explicit predictive interior provides.

A.4 Continuous-depth models and trajectory regularization

The dynamical-systems view of residual networks (E, 2017; Haber & Ruthotto, 2017; Lu et al., 2018) led to Neural ODEs (Chen et al., 2018) and their time-series variants, latent ODEs and neural CDEs (Rubanova et al., 2019; Kidger et al., 2020), which model continuous *time* rather than depth.

Exploding gradients in unrolled recurrences are a classical problem (Pascanu et al., 2013), and stability conditions for ResNets as ODEs were studied by Haber & Ruthotto (2017). Kinetic and Jacobian regularization (Finlay et al., 2020) and penalties on higher-order derivatives (Kelly et al., 2020) encourage straight, easily integrated trajectories. Our path term plays a similar role but acts on decoded quantiles rather than on the latent state. The budget conditioning uses feature-wise affine modulation (Perez et al., 2018) in the adaLN form of diffusion Transformers (Peebles & Xie, 2023), and register tokens stabilize attention (Darcet et al., 2024).

A.5 Flow matching, diffusion, and few-step transport

Diffusion and score-based models (Ho et al., 2020; Song et al., 2021) and flow matching with stochastic interpolants (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023) learn to transport a simple source to the data distribution. Straight paths reduce integration error (Liu et al., 2023; Tong et al., 2024). Few-step generation has been obtained through progressive distillation (Salimans & Ho, 2022), consistency models (Song et al., 2023), step-size-conditioned shortcut models (Frans et al., 2025), and mean flows (Geng et al., 2025). Several of these methods enforce agreement between trajectories of different step counts, a property our fixed-depth models lack (Section 5.3). They therefore suggest natural extensions, such as an endpoint-consistency loss across budgets. FlowTSFM differs from all of these models in what is transported: marginal quantiles of a deterministic forecaster, evolving along encoder depth under a supervised straight path, rather than samples evolving along generation time.

B Path Matching and Velocity Regression

This section proves Proposition 1 and extends it to the sampled-exit objective used in training. Throughout, the input, the number of steps K, and the scored set $\mathcal{M}$ are fixed. Every MSE averages over the same $|\mathcal{M}|Q$ position–quantile entries in normalized coordinates, and the source r is repeated at every position.

B.1 Proof of Proposition 1

Proof. Set $b = \mathrm{sg}(q^K)$ and $u = b - r$. By definition of the decoded velocities, $q^k = q^0 + \Delta t \sum_{i=0}^{k-1} v^i$. Subtracting $\pi^k = r + t_k u$ and using $t_k = k\Delta t$ gives

$$q^k - \pi^k = q^0 - r + \Delta t \sum_{i=0}^{k-1} (v^i - u). \tag{11}$$

Taking the mean squared value over scored entries proves Eq. 6. With $a = q^0 - r$ and $\bar{v}^{\,k} = k^{-1} \sum_{i<k} v^i$, the same identity gives

$$\mathrm{MSE}(q^k, \pi^k) = t_k^2 \,\mathrm{MSE}\left(\bar{v}^{\,k} + \frac{a}{t_k},\, u\right), \qquad 1 \le k \le K-1, \tag{12}$$

which reduces to the average-velocity identity of Proposition 1 when $q^0 = r$. Exact anchoring is needed only for this simplified form, not for Eq. 6 or for the bounds.

For the bounds, define $e^k = q^k - \pi^k$ for $k = 0, \dots, K$. Then $e^0 = q^0 - r$, and $e^K = 0$ in the forward pass because $\mathrm{sg}(q^K) = q^K$. Hence

$$\mathcal{R}_{\mathrm{path}} = \sum_{k=0}^{K-1} \mathrm{MSE}(e^k, 0), \qquad v^k - u = \frac{e^{k+1} - e^k}{\Delta t}. \tag{13}$$

Using $\mathrm{MSE}(A - B, 0) \le 2\,\mathrm{MSE}(A, 0) + 2\,\mathrm{MSE}(B, 0)$ at each depth,

$$\mathcal{R}_{\mathrm{vel}} = K^2 \sum_{k=0}^{K-1} \mathrm{MSE}(e^{k+1} - e^k, 0) \le 2K^2 \sum_{k=0}^{K-1} \left[\mathrm{MSE}(e^{k+1}, 0) + \mathrm{MSE}(e^k, 0)\right] \le 4K^2 \mathcal{R}_{\mathrm{path}}, \tag{14}$$

where the last step uses $e^K = 0$. This is the left inequality in Eq. 7.

Conversely, telescoping backwards from $e^K = 0$ gives $e^k = -\Delta t \sum_{i=k}^{K-1} (v^i - u)$. Applying Cauchy–Schwarz entrywise and averaging, $\mathrm{MSE}(e^k, 0) \le (\Delta t)^2 (K - k) \sum_{i=k}^{K-1} \mathrm{MSE}(v^i, u)$. Summing over k and exchanging the finite sums,

$$\mathcal{R}_{\mathrm{path}} \le \frac{1}{K^2} \sum_{i=0}^{K-1} \Big[\sum_{k=0}^{i} (K - k)\Big] \mathrm{MSE}(v^i, u) \le \frac{K+1}{2K} \mathcal{R}_{\mathrm{vel}}, \tag{15}$$

since $\sum_{k=0}^{i} (K - k) \le \frac{K(K+1)}{2}$. Both nonnegative regularizers therefore vanish exactly when $q^k = r + t_k(b - r)$ for every scored entry and every depth. □

B.2 Sampled exits

Training scores a uniform random subset $\mathcal{E}' \subseteq \{1, \dots, K-1\}$ of $m_{\mathrm{exit}} = 3$ interior exits. Each exit is selected with probability $p = \frac{m_{\mathrm{exit}}}{K-1}$, so by linearity of expectation

$$\mathbb{E}_{\mathcal{E}'} \Big[\sum_{k \in \mathcal{E}'} \mathrm{MSE}(q^k, \pi^k)\Big] = p \sum_{k=1}^{K-1} \mathrm{MSE}(q^k, \pi^k). \tag{16}$$

For the source and path terms of Eq. 5, let $\mathcal{R}_{\mathrm{sampled}} = \lambda_0 \,\mathrm{MSE}(q^0, r) + \lambda_{\mathrm{qf}} \sum_{k \in \mathcal{E}'} \mathrm{MSE}(q^k, \pi^k)$. With $c_- = \min\{\lambda_0, p\lambda_{\mathrm{qf}}\}$ and $c_+ = \max\{\lambda_0, p\lambda_{\mathrm{qf}}\}$,

$$\frac{c_-}{4K^2} \mathcal{R}_{\mathrm{vel}} \;\le\; \mathbb{E}_{\mathcal{E}'}[\mathcal{R}_{\mathrm{sampled}}] \;\le\; c_+ \frac{K+1}{2K} \mathcal{R}_{\mathrm{vel}}, \tag{17}$$

so exit sampling preserves the zero-loss characterization in expectation. With our settings ($\lambda_0 = 1$, $\lambda_{\mathrm{qf}} = 0.3$, $K = 12$), $p = 3/11$.

B.3 SCOPE OF THE RESULT

For the update of Eq. 1, the decoded velocities are exactly

$$v^k = \frac{D_\phi\big(h^k + \Delta t\, F_\theta(h^k, e_t(t_k), e_{\text{depth}}(K))\big) - D_\phi(h^k)}{\Delta t}, \tag{18}$$

with no assumption that the decoder is linear and no continuous-depth approximation. Conditional flow matching regresses a learned field at prescribed interpolants of a sample pair (Lipman et al., 2023). Here, the states come from the model's own rollout, and the endpoint is its own detached prediction. The proposition therefore gives an integrated-velocity reading of the position loss at a fixed depth. It says nothing about gradients, about the minimizers of the full objective, or about agreement between the endpoints of different budgets; the last point is studied empirically in Section 5.3.

Table 4: **Directional trajectory study.** CosMean aligns local decoded updates with their remaining terminal displacement. Dataset-macro averages weight 55 GIFT-Eval datasets equally; path-weighted averages weight valid test trajectories equally. Higher is better. The common-quantile probing protocol is detailed in Appendix C.

Model	Parameters	Dataset macro ↑	Path weighted ↑
FlowTSFM	38.8M	**0.919**	**0.933**
Chronos-2	119.5M	0.350	0.317

C CosMean: Definition and Probing Protocol

CosMean (Eq. 10) averages, over steps, the cosine between each decoded update and the remaining displacement to the final forecast, in the spirit of the straightness diagnostics of rectified flow (Liu et al., 2023):

$$\mathrm{CosMean}(q) = \frac{1}{K-1} \sum_{k=0}^{K-2} \frac{\langle q^{k+1} - q^k,\, q^K - q^k \rangle}{\|q^{k+1} - q^k\|_2 \, \|q^K - q^k\|_2}. \tag{19}$$

Inner products are taken over the flattened scored position–quantile entries. The last update is omitted because its cosine is trivially one. Terms whose denominator falls below 10^{-8} are set to 0. CosMean measures direction only: it is insensitive to speed, calibration, and accuracy, which are evaluated separately.

Protocol Both models are restricted to the nine quantile levels they share, $\{0.1, \ldots, 0.9\}$. For FlowTSFM, we decode all $K = 12$ states of the base checkpoint (Table 2) with its shared decoder. For Chronos-2, we apply its output head to the hidden state after each of its 12 encoder blocks, a logit-lens-style probe (nostalgebraist, 2020). We report two aggregates on GIFT-Eval: a dataset-macro average, which weights the 55 datasets equally, and a path-weighted average, which weights every test series equally.

Results Table 4 gives 0.919 (dataset-macro) and 0.933 (path-weighted) for FlowTSFM, against 0.350 and 0.317 for Chronos-2. As discussed in Section 5.2, FlowTSFM is trained toward this geometry, so we read the gap as evidence that the objective shapes the interior as intended, not as a measure of forecasting quality. Isolating the separate roles of weight sharing and of the path term would require a loss-off model that trains stably (Appendix H).

D Evaluation Protocol and Training Details

D.1 Benchmarks and baselines

All scores use Seasonal-Naive normalization per task followed by geometric-mean aggregation, on GIFT-Eval (55 datasets, 97 configurations) and TIME (98 configurations) (Aksu et al., 2024; Qiao et al., 2026). Baseline scores and parameter counts come from the official leaderboards; for Chronos-2, we use its official GIFT-Eval submission over the 97 configurations shared with Seasonal Naive and the TIME benchmark aggregate (Ansari et al., 2025). Baselines were therefore not retrained under a common data and compute budget. Our pretraining data are GIFT-Eval-Pretrain, which is designed to exclude the GIFT-Eval test series, and synthetic series.

D.2 Which checkpoint produced which result

- **Accuracy (Table 2) and CosMean (Table 4):** the base model, $K = 12$, batch size 1,024.
- **Fixed-depth study (Figure 4, Tables 6 and 9, Appendix F):** five checkpoints with $K_{\text{train}} \in \{4, 6, 8, 10, 12\}$, batch size 128, otherwise identical to the base model. Each is evaluated at all five budgets on both benchmarks (100 aggregate scores), and every run covers all 97 GIFT-Eval and 98 TIME configurations.
- **Randomized depth (Table 3, Appendix G):** one checkpoint with $K \sim \text{Unif}\{3, \ldots, 15\}$, batch size 128.
- **Stability ablation (Section 5.2, Appendix H):** the 12-step configuration of the fixed-depth study with $\lambda_{\text{qf}} = 0$.

D.3 Hyperparameters

Table 5: Training configurations. The three configurations differ only in the rows marked †.

Setting	Base	Fixed-depth study	Randomized depth
Recurrent steps $K^{\dagger}$	12	$\{4, 6, 8, 10, 12\}$	$\sim \text{Unif}\{3, \ldots, 15\}$
Batch size†	1,024	128	128
Context length / patch size	8,192 / 16		
Width / attention heads / MLP width	1,024 / 16 / 4,096		
Pre-recurrent layers / registers	1 / 4		
Output quantiles	99		
Parameters	38,845,536		
Data (real / kernel-synthetic)	50% / 50%		
Masking / masked-patch rate	continuous patches / 40%		
Optimizer / precision	AdamW / bfloat16		
Training steps	100,000		
Peak / final learning rate	3×10^{-4} / 10^{-5}		
Warm-up steps / weight decay	5,000 / 0.1		
Gradient clipping (max norm)	1.0		
Sampled interior exits m_{exit}	3		
λ_{ds} / λ_{qf} / λ_0	0.5 / 0.3 / 1		
Hardware	4× NVIDIA H200		

E DETAILED ARCHITECTURE

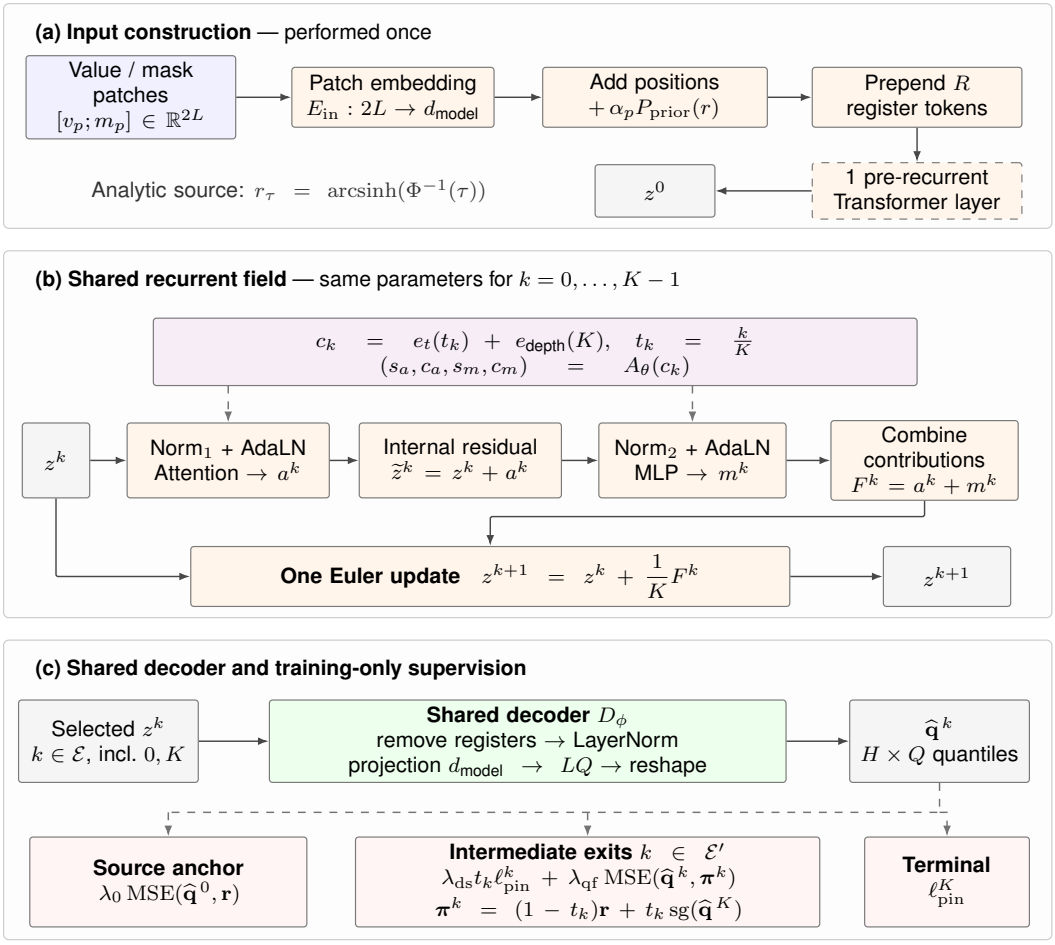


Figure 5: **Detailed FlowTSFM computation.** Input construction and the pre-recurrent layer run once. Attention and MLP together form one shared update; the residual connection inside the block is not an extra Euler step. The three losses at the bottom are used only during training.

Input construction For a value patch v_p and mask patch m_p of length $L = 16$, the input embedding maps $[v_p; m_p] \in \mathbb{R}^{2L}$ to width d_{model}. We add learned positional embeddings and $\alpha_p P_{\text{prior}}(r)$, a learned projection of the source weighted by the fraction α_p of the patch that lies in the forecast window $\mathcal{M}$. The source thus enters the prediction positions from the start. Four learned register tokens are prepended, and the pre-recurrent Transformer layer produces h^0.

Shared block and adaptive normalization At each step, the depth and budget embeddings produce four affine controls,

$$c_k = e_t(t_k) + e_{\text{depth}}(K), \qquad (s_a, c_a, s_m, c_m) = A_\theta(c_k), \tag{20}$$

with $\operatorname{mod}(u; s, c) = u \odot (1 + c) + s$. The block computes

$$\begin{aligned} a^k &= \operatorname{MHA}_\theta\big(\operatorname{mod}(\operatorname{N}_1(h^k); s_a, c_a)\big), \qquad \widetilde{h}^k = h^k + a^k, \\ m^k &= \operatorname{MLP}_\theta\big(\operatorname{mod}(\operatorname{N}_2(\widetilde{h}^k); s_m, c_m)\big), \qquad F^k = a^k + m^k, \qquad h^{k+1} = h^k + \frac{1}{K} F^k. \end{aligned} \tag{21}$$

Attention is bidirectional multi-head self-attention with a padding mask. The MLP maps $d_{\text{model}} \to 4d_{\text{model}} \to d_{\text{model}}$ with GeLU. All block parameters, including A_θ, are shared across steps, and the block has no gates. Only the last line of Eq. 21 updates the recurrent state.

Shared decoder The decoder removes the register positions, applies a final LayerNorm, and projects each patch from d_{model} to LQ values, reshaped into the $T \times Q$ quantile array. Input embedding and output projection are residual projection blocks. At deployment only the final state is decoded; intermediate readouts are used for training and for the diagnostics of Section 5.2.

Training and inference Training unrolls K steps and applies the terminal pinball loss, the source anchor, and the two interior losses at the sampled exits (Eq. 5). Running with a different number of steps recomputes the grid, the step size, and the budget embedding (Section 3.4); it does not truncate the training grid.

Table 6: **Matched training and inference depth.** Lower is better. Scores are rounded to three decimal places; bold indicates the best recorded diagonal score in each column. All models use the shared-block architecture.

	GIFT-Eval (97 configurations)		TIME (98 configurations)	
$K_{\text{train}} = K_{\text{test}}$	MASE	CRPS	MASE	CRPS
4	0.761	0.520	0.719	0.599
6	0.754	0.513	0.711	0.594
8	0.749	0.511	0.716	0.594
10	0.757	0.518	0.707	0.593
12	**0.743**	**0.506**	**0.698**	**0.582**

Table 9: **Relative error increase from budget mismatch (%).** Entries summarize Eq. 9. The median uses all 20 off-diagonal cells; directional means use ten cells each. No metric is averaged across benchmarks.

Benchmark	Metric	Off-diagonal median	Mean: $K_{\text{test}} < K_{\text{train}}$	Mean: $K_{\text{test}} > K_{\text{train}}$
GIFT-Eval	MASE	11.082	29.042	8.206
GIFT-Eval	CRPS	14.488	29.960	10.766
TIME	MASE	9.828	30.292	7.464
TIME	CRPS	13.117	26.289	9.962

F Fixed-Depth Study: Complete Results

This section reports all 100 scores of the fixed-depth study summarized in Section 5.3. Rows are training depths and columns are inference depths. Bold entries are row minima, which always lie on the diagonal. Figure 4 shows the CRPS grids as heatmaps.

Table 7: Normalized MASE for every pair of training and inference depths.

GIFT-Eval

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	**0.761**	0.843	0.798	0.880	0.822
6	0.913	**0.754**	0.813	0.839	0.774
8	1.003	0.812	**0.749**	0.805	0.787
10	1.142	0.874	0.808	**0.757**	0.820
12	1.378	1.046	0.854	0.840	**0.743**

TIME

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	**0.719**	0.786	0.743	0.832	0.769
6	0.845	**0.711**	0.758	0.785	0.729
8	0.963	0.776	**0.716**	0.766	0.748
10	1.102	0.840	0.760	**0.707**	0.769
12	1.272	1.002	0.826	0.802	**0.698**

Table 8: Normalized CRPS for every pair of training and inference depths.

GIFT-Eval

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	**0.520**	0.592	0.556	0.619	0.573
6	0.631	**0.513**	0.573	0.593	0.537
8	0.669	0.569	**0.511**	0.562	0.543
10	0.774	0.608	0.556	**0.518**	0.567
12	0.956	0.716	0.583	0.583	**0.506**

TIME

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	**0.599**	0.672	0.633	0.701	0.652
6	0.693	**0.594**	0.657	0.678	0.617
8	0.745	0.658	**0.594**	0.651	0.632
10	0.851	0.695	0.649	**0.593**	0.659
12	1.001	0.799	0.671	0.671	**0.582**

Penalty for a budget mismatch The next tables report δ_{ij} (Eq. 9) in percent, the increase in error when a checkpoint runs at a budget other than its own. Table 9 summarizes them.

Table 10: Relative MASE increase from each checkpoint's matched-depth score (%).

GIFT-Eval

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	0.000	10.820	4.970	15.646	8.046
6	21.136	0.000	7.855	11.344	2.667
8	33.841	8.357	0.000	7.449	4.993
10	50.845	15.386	6.749	0.000	8.267
12	85.428	40.770	14.841	13.065	0.000

TIME

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	0.000	9.295	3.339	15.709	6.929
6	18.839	0.000	6.594	10.361	2.446
8	34.427	8.265	0.000	6.869	4.356
10	55.826	18.792	7.438	0.000	8.739
12	82.397	43.564	18.463	14.908	0.000

Table 11: Relative CRPS increase from each checkpoint's matched-depth score (%).

GIFT-Eval

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	0.000	13.846	6.942	18.981	10.115
6	23.063	0.000	11.863	15.610	4.800
8	30.933	11.250	0.000	9.861	6.300
10	49.267	17.349	7.256	0.000	9.340
12	88.781	41.379	15.129	15.189	0.000

TIME

$K_{\text{train}} \backslash K_{\text{test}}$	4	6	8	10	12
4	0.000	12.066	5.607	17.006	8.879
6	16.779	0.000	10.630	14.168	3.976
8	25.493	10.783	0.000	9.756	6.420
10	43.493	17.229	9.390	0.000	11.109
12	71.959	37.285	15.223	15.258	0.000

F.1 PER-CHECKPOINT INFERENCE SWEEPS

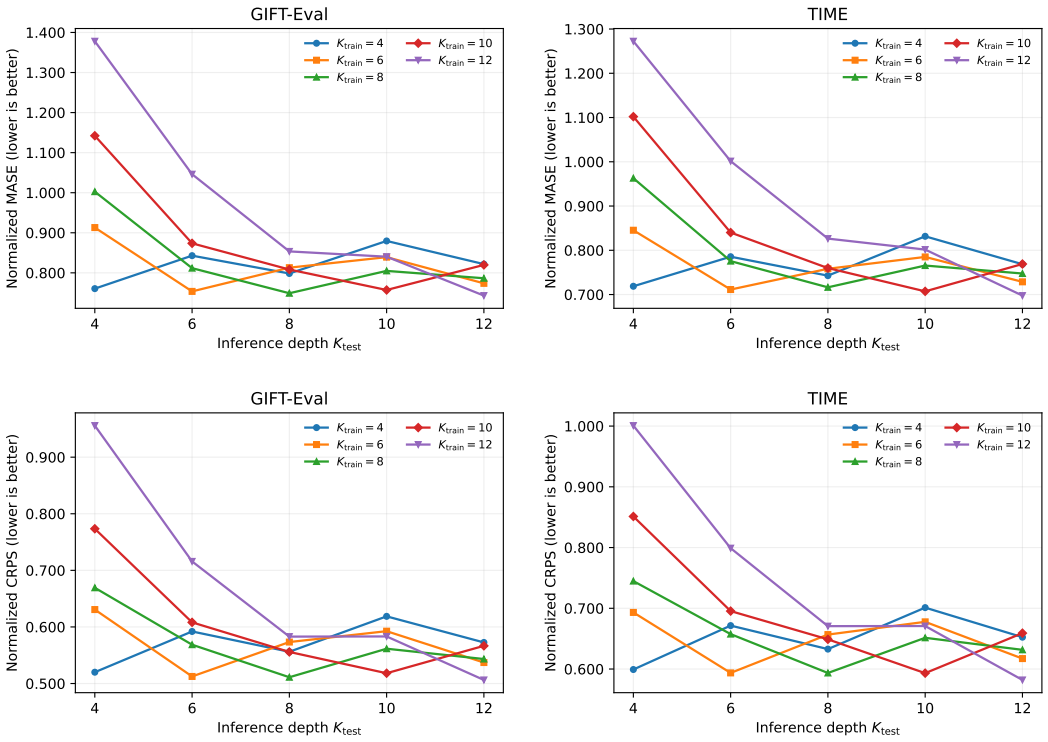


Figure 6: **Each fixed-depth checkpoint run at every budget.** Each curve is one row of the grids above and reaches its minimum at its own training depth. Compare with the randomized-depth checkpoint (Table 3), whose accuracy improves with depth.

Moving toward its training depth helps a badly mismatched checkpoint, but going beyond it never beats the matched result.

F.2 SHALLOW SPECIALISTS VERSUS A SHORTENED DEEP MODEL

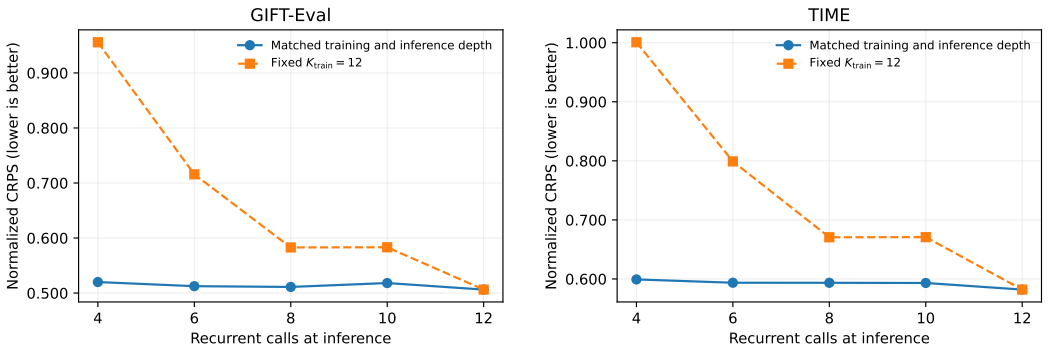


Figure 7: **Separately trained shallow models versus a shortened deep model** (normalized CRPS). Solid: checkpoints run at their training depth. Dashed: the 12-step checkpoint run with fewer steps.

The 4-step specialists score 0.761/0.520 (MASE/CRPS, GIFT-Eval) and 0.719/0.599 (TIME), which is 2.3%/2.7% and 3.0%/3.0% worse than the 12-step specialists. The 12-step checkpoint run with 4 steps scores 1.378/0.956 and 1.272/1.001, which is 85.4%/88.8% and 82.4%/72.0% worse. With

fixed-depth training, a cheaper model must be trained at its target depth; Appendix G shows that randomized-depth training removes this need.

G RANDOMIZED-DEPTH CHECKPOINT

The randomized-depth checkpoint follows Table 5. At every training step, it draws K uniformly from $\{3, \ldots, 15\}$ and applies the loss of Eq. 5 to a rollout of that length, whose path target uses the rollout's own detached final forecast. We evaluate it with 3, 6, 9, 12, 15, and 18 steps; 18 lies outside the training range. The scores are in Table 3. Table 12 expresses them relative to the two alternatives at the budgets where both exist.

Table 12: Relative error of the single randomized-depth checkpoint (%, negative is better) with respect to (a) the specialist trained at the same budget and (b) the 12-step checkpoint run at that budget. MASE / CRPS.

	vs. specialist		vs. shortened 12-step	
Steps	GIFT-Eval	TIME	GIFT-Eval	TIME
6	-0.8 / $+0.8$	-0.6 / -0.8	-28.5 / -27.8	-29.4 / -26.3
12	$+0.4$ / $+2.0$	$+0.4$ / $+0.5$	$+0.4$ / $+2.0$	$+0.4$ / $+0.5$

At 12 steps the shortened model and the specialist coincide, which explains the identical columns. On both benchmarks, accuracy improves from 3 to 9 steps, remains approximately constant through 15, and degrades at 18. These results support test-time scaling within the sampled training range, but not reliable extrapolation beyond it. The following subsection examines the sensitivity of the learned budget embedding as a possible contributor to this limitation.

H TRAINING WITHOUT THE PATH TERM

Setup The ablation uses the 12-step configuration of the fixed-depth study (Table 5, batch size 128, gradient clipping at 1.0) and sets $\lambda_{\mathrm{qf}} = 0$ in Eq. 5. The terminal pinball loss, the deep-supervision loss, and the source anchor are unchanged. We ran this ablation twice, and both runs diverged. With the path term, the same configuration trains stably for 100,000 steps.

Representative run Table 13 shows the training log of one run, recorded every 100 steps. The logged path loss is zero throughout, which confirms the intervention. The gradient norm, measured before clipping, grows from about 1 at step 1,700 to 3.7×10^4 at step 2,800 and 1.5×10^{14} at step 2,900, and becomes infinite at step 3,000. Clipping keeps the updates bounded for a while, so the losses stay finite, and finite gradient norms briefly reappear around steps 5,900–6,500 and 18,000. From step 22,700 on, all losses and the gradient norm are NaN.

Table 13: Training log of a representative run without the path term ($\lambda_{\mathrm{qf}} = 0$, 12 steps). Gradient norms are measured before clipping. These are training diagnostics, not benchmark scores.

Step	Gradient norm	Total loss	Terminal loss
1,700	1.027	0.458	0.257
2,600	358.8	0.744	0.415
2,800	3.695×10^4	0.779	0.433
2,900	1.533×10^{14}	1.164	0.691
3,000	∞	1.944	1.133
6,500	9.022×10^{13}	0.836	0.466
22,600	∞	0.924	0.509
22,700	NaN	NaN	NaN
100,000	NaN	NaN	NaN

Interpretation With deep supervision alone, each exit is pulled toward the target on its own, and nothing constrains how the shared block moves the state from one exit to the next. Because the same block is applied twelve times, a small increase in its gain compounds across steps, which is consistent with the sudden explosion of the gradient norm. The path term assigns every exit a position on the line from the source to the final forecast, and by Proposition 1 it penalizes the velocity error accumulated along the rollout. In our experiments, this is enough to keep training stable.

H.1 SENSITIVITY OF THE BUDGET EMBEDDING

We probe how the learned conditioning changes with the requested recurrent budget, separately from forecasting accuracy. Equation 1 uses both the learned budget embedding $e_{\mathrm{depth}}(K)$ and the explicit step size $\Delta t = \frac{1}{K}$. The diagnostic examines the former and its projection into adaptive layer-normalization (AdaLN) controls. It does not measure changes in the recurrent hidden states or the vector field along a rollout.

Checkpoint protocol Both checkpoints are analyzed at the recorded training step $100{,}000$. The fixed-depth checkpoint uses $K = 12$, an effective batch size of 128 (64×2), and training from scratch. The randomized-depth checkpoint samples $K \sim \mathrm{Unif}\{3, \ldots, 15\}$, uses a batch size of 256 (256×1), and is warm-started from a fixed-depth checkpoint at step $60{,}000$. Both diagnostic run configurations record $10{,}000$ warm-up steps. These settings are specific to the probed checkpoints; they should not be conflated with the otherwise-matched, batch-128 depth study described in Section 4. In particular, the comparison below characterizes budget dependence within each checkpoint, but does not isolate the effect of depth randomization from batch size or initialization. The training-step labels do not establish matched cumulative training compute.

Matched-progress probing Let $e(K) = e_{\mathrm{depth}}(K)$ and write the summed conditioning and its AdaLN projection as

$$S(t, K) = e_t(t) + e(K), \qquad m_{\mathrm{ada}}(t, K) = A_\theta\big(S(t, K)\big), \tag{22}$$

where m_{ada} concatenates the shift and scale controls. At fixed t, the time embedding cancels exactly:

$$S(t, K) - S(t, 12) = e(K) - e(12). \tag{23}$$

We evaluate the integer grid $\mathcal{K} = \{1, \ldots, 32\}$ and report finite differences relative to $K = 12$ at $t = 0.5$. The relative-change columns use the notation

$$\begin{aligned} \Delta e/e &= 100\, \frac{\|e(K) - e(12)\|_2}{\|e(12)\|_2}, \\ \Delta e/S &= 100\, \frac{\|e(K) - e(12)\|_2}{\|S(0.5, 12)\|_2}, \\ \Delta m/m &= 100\, \frac{\|m_{\mathrm{ada}}(0.5, K) - m_{\mathrm{ada}}(0.5, 12)\|_2}{\|m_{\mathrm{ada}}(0.5, 12)\|_2}. \end{aligned} \tag{24}$$

Holding progress fixed isolates the explicit budget dependence of the conditioning. It does not imply that actual rollouts visit the same hidden states or the same discrete progress grid.

Embedding geometry To separate a budget-independent component from variation across budgets, define

$$\bar{e} = \frac{1}{32} \sum_{K \in \mathcal{K}} e(K), \qquad \mathrm{dev}(K) = e(K) - \bar{e}. \tag{25}$$

Table 14 reports the diagnostic's centered RMS, relative centered magnitude, and leading-direction energy. The minimum pairwise cosine is 0.999929 for the fixed-depth checkpoint, compared with 0.914193 for the randomized-depth checkpoint. The latter has substantially more variation, with 96.4% of the centered energy in one direction; its leading coordinate is monotone in K up to 15. The fixed-depth profile is non-monotone. These are properties of the measured checkpoints, not a claim that training at one depth necessarily produces a constant embedding at all other depths.

Table 14: **Budget-embedding geometry over $K \in \{1, \ldots, 32\}$.** Centered magnitudes and energy fractions retain the diagnostic's reported aggregation convention. The last row describes ordering of the leading coordinate. Checkpoint training settings differ as stated in the protocol.

Statistic	Fixed depth $K = 12$	Randomized depth $K \in \{3, \ldots, 15\}$
Minimum pairwise cosine	0.999929	0.914193
RMS of dev	8.02×10^{-3}	5.52×10^{-1}
Relative centered magnitude	0.0030	0.1852
Energy in leading direction	94.5%	96.4%
Monotone ordering in K	No	Yes, for $K \leq 15$

Sensitivity of conditioning and AdaLN controls Table 15 shows that the fixed-depth checkpoint has small relative changes at each tested budget, all below 0.4%. For $K : 12 \to 3$, the summed conditioning changes by 0.319% in the fixed-depth checkpoint and 40.3% in the randomized-depth checkpoint; the corresponding AdaLN-control changes are 0.360% and 64.1%. Budget dependence is therefore visible not only in the embedding but also in the controls used by the recurrent block. These relative changes have different reference norms, so their ratios should not be interpreted as operator gains or changes in forecasting error.

Saturation outside the training range For the randomized-depth checkpoint, compare spans with the same width in K, normalizing each displacement by the embedding norm at the lower endpoint:

$$D(a, b) = 100\, \frac{\|e(b) - e(a)\|_2}{\|e(a)\|_2}, \qquad b - a = 5. \tag{26}$$

Table 16 shows displacements of 19.5% and 9.1% over $5 \to 10$ and $10 \to 15$, respectively, but only 2.0% over $15 \to 20$ and 1.2% over $20 \to 25$. The smaller changes above 15 indicate flattening rather than continued separation at the same rate as within the training range. The leading coordinate spans 50.7 units over $K \in \{1, \ldots, 15\}$, changes by 2.9 units between 12 and 15, and spans only 1.1 units over $K \in \{18, \ldots, 32\}$. The first of these ranges also includes the untrained budgets 1 and 2.

Table 15: **Relative changes (%) from $K = 12$ at $t = 0.5$.** The three quantities are defined in Eq. 24. The time embedding is held fixed. $K = 20$ lies outside the randomized training range.

	Fixed depth ($K = 12$)			Randomized depth (3–15)		
K	$\frac{\Delta e}{e}$	$\frac{\Delta e}{S}$	$\frac{\Delta m}{m}$	$\frac{\Delta e}{e}$	$\frac{\Delta e}{S}$	$\frac{\Delta m}{m}$
3	0.345	0.319	0.360	42.9	40.3	64.1
6	0.136	0.126	0.094	23.7	22.2	34.5
15	0.003	0.003	0.004	4.2	3.9	7.6
20	0.059	0.054	0.033	5.7	5.3	12.3

Table 16: **Five-step embedding displacements for the randomized-depth checkpoint.** Relative changes are normalized by the embedding norm at the lower endpoint. The training range ends at $K = 15$.

Span	$\|e(b) - e(a)\|_2$	$D(a, b)$	Location
$5 \to 10$	18.54	19.5%	Within training range
$10 \to 15$	7.22	9.1%	Within training range
$15 \to 20$	1.52	2.0%	Boundary to outside
$20 \to 25$	0.93	1.2%	Outside training range
$27 \to 32$	1.08	1.4%	Outside training range

The relative conditioning distance $\frac{\Delta e}{S}$ from the $K = 12$ reference is also non-monotone beyond the training range: it is 3.9% at $K = 15$, 5.3% at $K = 20$, and 3.8% at $K = 32$. Embedding norms vary little across these larger budgets: 74.2, 74.1, 74.4, 75.1, and 75.6 at $K = 15, 18, 20, 24, 32$, respectively. Similar distances from one reference do not establish identical conditioning vectors, nor do they show that an out-of-range budget is mapped to a particular in-range budget.

Interpretation and scope The step size remains budget-dependent even when the embedding varies little: changing K from 12 to 3 changes Δt from $\frac{1}{12}$ to $\frac{1}{3}$. For a fixed hidden state and progress, this positive scalar rescales the field evaluation without changing its direction. Actual changes in the rollout budget additionally alter the number of updates, the progress grid, and the states visited. A weakly varying budget embedding therefore does not imply that all rollouts execute identical dynamics or produce identical forecasts.

Together, the measurements identify limited out-of-range sensitivity of the learned budget signal in the probed randomized-depth checkpoint. This is consistent with the decline in accuracy beyond the sampled range in Table 3, and suggests a possible explanation for limited extrapolation. It is not a causal intervention on the embedding, and the diagnostic checkpoint should not be identified with an otherwise-matched accuracy run without verifying its training manifest. Establishing that embedding saturation causes the accuracy decline would require a controlled comparison. The present measurements also do not determine whether additional training, model capacity, or a different budget parameterization would improve extrapolation.

I FURTHER DIAGNOSTICS

Two further diagnostics would complement CosMean. *Straightness*, $S_{\mathrm{rel}} = \sum_k \Delta t_k \frac{\|v_k - c\|_2^2}{\|c\|_2^2 + \varepsilon}$ with chord $c = q^K - q^0$, penalizes both curvature and uneven speed along depth, which a directional cosine does not capture. The *cross-budget endpoint discrepancy*, $E(n) = \frac{\|q^{(n)} - q^{(\mathrm{ref})}\|_2^2}{\|q^{(\mathrm{ref})} - q^0\|_2^2 + \varepsilon}$, compares the final forecasts of one checkpoint run with n steps and with a reference number of steps. It would measure directly how closely randomized-depth training aligns the endpoints of different budgets. Quantile-crossing rates and empirical interval coverage per budget would complete the probabilistic evaluation, and latency and memory measurements would turn our step counts into wall-clock costs. We leave these measurements to future work.