TEACHERS ARE NOT EQUAL: TASK-FLOW CONDITIONAL CROSS-LAYER KNOWLEDGE DISTILLATION FOR EFFICIENT FLOW-BASED VLA MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models based on flow matching are expensive to deploy due to repeated Transformer computation at inference. One solution is to distill knowledge from a teacher VLA to a lightweight student VLA. Existing methods select a fixed teacher layer for distillation, but this may be suboptimal because layer informativeness varies with task instructions and flow stages, limiting student capacity. To address this, we introduce **T**ask-**F**low **C**onditional cross-layer **K**nowledge **D**istillation (`TFC-KD`), a framework that adaptively selects teacher layers conditioned on task instructions and flow stages. For each retained student layer, we first identify several candidate teacher layers around its initially aligned counterpart. We then use task instructions to modulate which teacher layers provide stronger supervision. Crucially, we divide the continuous flow process into three stages, allowing the preferred teacher layers to shift as action generation progresses. Task instructions and flow stages are combined to assign weights to candidate teacher layers, whose weighted attention maps over vision-language tokens serve as distillation targets. Finally, we impose an ordering constraint to keep teacher-layer selections roughly aligned with student depth while permitting local adaptation across task instructions and flow stages. On LIBERO, the trained 8-layer student achieves 96.72% success, outperforming the Shallow-π baseline by 1.57%. When further compressed to 6 layers, `TFC-KD` incurs only a 0.76% success drop relative to the teacher ($96.28\% \rightarrow 95.52\%$), while reducing deployed parameters by 44.37% and FLOPs by 58.07%, with a speedup of $2.30\times$ inference on one A800 GPU. Our code is available at **Here**.

1 INTRODUCTION

Flow-based Vision-Language-Action (VLA) models (Black et al., 2025b;a; NVIDIA et al., 2025; Deng et al., 2025; Shukor et al., 2025) have achieved promising performance in general-purpose robotic manipulation by combining visual perception, language conditioning, and continuous action generation within a single policy. Flow matching generates continuous actions by progressively transforming noise into action trajectories (Lipman et al., 2023). Recent models such as π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), GR00T N1 (NVIDIA et al., 2025), GraspVLA (Deng et al., 2025), and SmolVLA (Shukor et al., 2025) adopt flow-based action generation for continuous robot control. Because action generation requires repeated computations of Transformer stacks over multiple flow steps, model depth contributes to the computational cost of every generation step, making efficient deployment difficult when low latency and frequent closed-loop control are required (Jeon et al., 2026; Yue et al., 2024; Reuss et al., 2025).

Existing approaches reduce this deployment cost from different directions. Consistency Policy (Prasad et al., 2024) reduces the cost of iterative action generation through consistency distillation, but it does not directly reduce the depth of the policy network. DeeR-VLA (Yue et al., 2024) dynamically varies the active model depth during inference, reducing computation adaptively but resulting in input-dependent execution rather than a single fixed shallow policy. FLOWER (Reuss et al., 2025) improves VLA efficiency through architectural design and model scaling, rather than directly compressing a fixed model into a shallower network. In contrast, structural depth compression directly removes Transformer layers. It therefore reduces the computation of every flow step while retaining a fixed inference graph for all inputs. Shallow-π (Jeon et al., 2026) follows this direction

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.

and transfers action supervision, teacher velocity, and intermediate attention to a shallow student. However, its intermediate attention supervision is taken from a fixed teacher layer, regardless of the current task instruction or flow stage. This fixed choice may provide less suitable supervision when different task instructions or flow stages benefit from different intermediate teacher representations.

More broadly, knowledge distillation has shown that intermediate teacher representations can provide useful supervision to compact students (Hinton et al., 2015; Romero et al., 2015; Zagoruyko & Komodakis, 2017). Several methods further relax fixed teacher–student layer correspondence. SemCKD (Chen et al., 2021a) learns soft associations between teacher and student layers; SAKD (Song et al., 2022) changes distillation locations across samples and training stages; TED (Liang et al., 2023) introduces task-aware layer-wise transfer; and C2KD (Chen et al., 2025) dynamically routes cross-layer supervision. These studies show that the best source of intermediate supervision need not be fixed. However, they do not consider a defining property of flow-based VLA policies: the same action is progressively refined through different flow stages. In this setting, the useful teacher layer may change not only with the task, but also as action generation progresses. Our gradient-based analysis confirms that the preferred teacher layer varies across task instructions, flow stages, and student layers. This motivates a question: *can we adapt teacher-layer supervision during training while keeping the deployed student completely static?* Appendix D further discusses generalist robot policies, efficient VLA inference, structural compression, and adaptive cross-layer distillation.

To address this problem, we propose **T**ask-**F**low **C**onditional cross-layer **K**nowledge **D**istillation (`TFC-KD`). The key idea is to keep the student architecture fixed while adapting the teacher layers used for supervision during training. In particular, for each retained student layer, we first select several nearby teacher layers around its initial teacher–student alignment. We then use the task instruction to adjust which teacher layers should provide more supervision. More importantly, we divide the continuous flow process into three stages, allowing the preferred teacher layers to change as action generation progresses. We then combine the task instruction and flow stage to weight the candidate teacher layers and use their weighted attention maps as the distillation target. Finally, we add an extra ordering constraint so that the selected teacher layers remain roughly consistent with student depth. All adaptive routing is used only during training and is removed after distillation, leaving a fixed shallow policy for deployment with no additional routing overhead.

In summary, the main contribution includes: 1) *New Observation.* We identify that the preferred teacher layer for a given student layer varies systematically across task instructions and flow stages under a gradient-compatibility criterion. 2) *Novel Methodology.* We propose `TFC-KD`, which uses a lightweight router to assign task- and flow-dependent mixture weights to adjacent teacher layers for each student layer, distilling the resulting weighted attention targets while removing all routing components after training. 3) *Comprehensive Validation.* Across three training seeds, `TFC-KD` improves the matched 8-layer Shallow-π baseline by 1.57%. With a 6-layer student, it remains within 0.76% of the teacher while achieving a measured $2.30\times$ A800 inference speedup. The gains further transfer across backbones, perturbations, compression depth, and cross-benchmark evaluation.

2 Related Work

Generalist and Generative Robot Policies. Language-conditioned robot policies range from multi-task imitation to large generalist models (Jang et al., 2022; Brohan et al., 2023; Zitkovich et al., 2023; Driess et al., 2023; Li et al., 2024; Ghosh et al., 2024; Kim et al., 2025). Generative policies use action chunking, diffusion, and flow matching for continuous control (Zhao et al., 2023; Ho et al., 2020; Chi et al., 2023; Liu et al., 2025; Lipman et al., 2023). In particular, π_0 and $\pi_{0.5}$ combine pretrained vision-language representations with flow-based action generation, requiring repeated evaluations of deep Transformer stacks (Black et al., 2025b;a).

Efficient VLA Inference and Compression. Efficient robot policies reduce inference cost through consistency distillation, adaptive depth, or architectural design (Prasad et al., 2024; Yue et al., 2024; Reuss et al., 2025). Our closest baseline, Shallow-π (Jeon et al., 2026), reduces Transformer depth in flow-based VLAs and distills action-to-vision-language attention at a fixed middle Transformer layer. We reproduce its L8 configuration under a unified pipeline and investigate whether teacher-layer supervision should instead depend on the task instruction and flow stage.

Knowledge Distillation and Adaptive Supervision. Knowledge distillation transfers teacher information to compact students, including through intermediate representations and attention (Hinton et al., 2015; Gou et al., 2021; Romero et al., 2015; Zagoruyko & Komodakis, 2017). Adaptive

methods learn soft layer associations, vary distillation locations, incorporate task awareness, or dynamically route cross-layer supervision (Chen et al., 2021a; Song et al., 2022; Liang et al., 2023; Chen et al., 2025). These methods do not condition teacher-layer selection on the flow stage of iterative action generation in flow-based VLAs. `TFC-KD` determines mixture weights over adjacent teacher layers from the student layer, task instruction, and flow stage. Removing the router after distillation preserves a fixed depth-compressed inference architecture. See Appendix D for details.

3 PRELIMINARIES

Flow-Based Action Generation. We denote the input images/videos, language instruction, and robot state as I, p, and s, respectively, and collectively refer to them as the observation $\mathbf{o} = (I, p, s)$. Given an expert action chunk a, Gaussian noise ϵ, and flow time $t \in [0, 1]$, we construct the noisy action state $x_t = (1-t)a + t\epsilon$ with target velocity $u_t = \epsilon - a$. The policy network F_θ takes $(\mathbf{o}, x_t, t)$ as input and predicts the velocity $v_t = F_\theta(\mathbf{o}, x_t, t)$ toward u_t. A larger value of t corresponds to a noisier action state, representing an earlier stage in the action refinement process.

Shallow Flow-VLA Distillation. Following Shallow-π (Jeon et al., 2026), we employ a frozen dense teacher model F_{θ_T} and a compressed lightweight student model F_θ. For the same observation $\mathbf{o} = (I, p, s)$ and noisy action state x_t, the teacher predicts $v_t^T = F_{\theta_T}(\mathbf{o}, x_t, t)$. In addition to action supervision and teacher velocity matching, Shallow-π distills attention from action queries to vision-language tokens at a fixed intermediate Transformer layer. Our method builds upon this structural compression framework but allows the teacher layer that provides attention supervision to vary with the different student layers, tasks, and flow stages.

Problem Setup. Given a frozen teacher with L_T layers and a trainable student with $L_S (< L_T)$ layers, both models process the same observation $\mathbf{o} = (I, p, s)$. Conventional intermediate distillation assigns each student layer to a fixed teacher layer for supervision. We investigate a more general problem: *how to adaptively select the teacher layer for supervising each student layer based on the current task instruction and flow stage, while maintaining a fixed student architecture?*

4 THE PROPOSED METHOD

Framework Overview. This paper investigates efficient compression of flow-based VLA models, where a depth-compressed student may require different teacher-layer supervision across tasks and flow stages. To address this, we propose `TFC-KD`, which adapts teacher supervision during training while maintaining a fixed student architecture. Our framework consists of three components: (1) *Static Depth Compression and Local Teacher Candidates* constructs a shallow student and restricts each layer to a limited neighborhood of depth-aligned teacher layers; (2) *Task-Flow Conditional Routing* leverages the task instruction and flow stage to determine each candidate teacher layer's contribution; (3) *Conditional Cross-Layer Distillation* aggregates candidate teacher attentions according to routing weights and supervises the student layer with the resulting target. All routing components are applied only during training and removed after distillation, yielding a fixed depth-compressed policy for deployment. An overview is illustrated in Figure 1.

4.1 STATIC DEPTH COMPRESSION AND LOCAL TEACHER-LAYER CANDIDATES

Our first goal is to construct a shallow student with a fixed inference architecture while preserving the coarse layer progression of the dense teacher. Allowing each student layer to receive supervision from arbitrary teacher layers would discard this structural correspondence and make cross-layer matching unnecessarily unconstrained. We therefore initialize the student from depth-aligned teacher layers and restrict the supervision of each student layer to a small local neighborhood around its initial teacher layer. Specifically, for a dense teacher with L_T Transformer layers, we compress to a student with $L_S < L_T$ layers while keeping the vision encoder, hidden dimensions, and action input/output projections unchanged. We denote the student layers as $S_0, \ldots, S_{L_S-1}$ and the teacher layers as $T_0, \ldots, T_{L_T-1}$. For each student layer S_l, we define its depth-aligned teacher layer index by uniformly distributing student layers across the teacher depth:

$$c_l = \left\lfloor \frac{l \times (L_T - 1)}{L_S - 1} + 0.5 \right\rfloor. \tag{1}$$

Student layer S_l is initialized from T_{c_l}, preserving the coarse progression of representations through the dense model. To allow flexibility while maintaining structural correspondence, we define a local

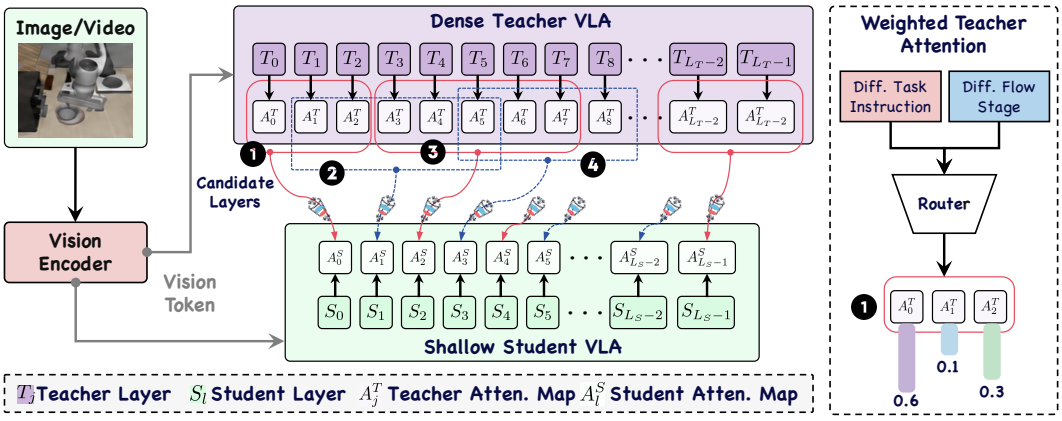

Figure 1: Overview of TFC-KD. Each shallow student layer is associated with a local neighborhood of teacher layers. Conditioned on the task instruction and flow stage, the router predicts mixture weights over these candidates and constructs a weighted teacher-attention for layer-wise distillation.

neighborhood of K candidate teacher layers around c_l:

$$\mathcal{C}_l = \{c_l - \lfloor K/2 \rfloor, \ldots, c_l + \lfloor K/2 \rfloor\} \cap \{0, \ldots, L_T - 1\}, \tag{2}$$

where the intersection ensures valid indices at the boundaries. Restricting supervision to $\mathcal{C}_l$ prevents unrestricted cross-depth matching while still allowing nearby teacher layers to provide alternative supervision. For routing, we index the candidate offsets relative to c_l by $\rho \in \{0, \ldots, K-1\}$, where offset ρ corresponds to teacher layer

$$j_{l,\rho} = c_l + \rho - \lfloor K/2 \rfloor. \tag{3}$$

We define the valid offset set for student layer l as

$$\mathcal{Q}_l = \{\rho : 0 \le j_{l,\rho} < L_T\}, \tag{4}$$

which excludes invalid candidates at the teacher boundaries. The local neighborhood determines where supervision may come from, but not which candidate should be preferred for a particular task instruction and flow stage. We address this next with task-flow conditional routing.

4.2 Task-Flow Conditional Routing

Task-Conditioned Residual. The local candidate set provides several plausible teacher layers for each student layer, but their usefulness can vary with the manipulation task. Different instructions emphasize different objects, spatial relations, and interaction requirements, which may favor different intermediate teacher representations. We therefore use the language instruction to produce a task-dependent correction to the teacher-layer preference. The router uses the language representation to encode the current task instruction. To prevent the attention-distillation objective from directly modifying the language features through the routing path, we pool the valid language-token features and stop gradients before passing the result to a lightweight task encoder:

$$e_p = \frac{\sum_k m_k h_k^{\text{lang}}}{\sum_k m_k}, \quad \widetilde{e}_p = \text{sg}(e_p), \quad r_\phi(p) = W_2\, \text{SiLU}(W_1 \widetilde{e}_p + b_1) + b_2, \tag{5}$$

where h_k^{lang} denotes the language token features from the student's language branch, $m_k \in \{0, 1\}$ is a binary mask that selects valid language tokens, $\text{sg}(\cdot)$ denotes the stop-gradient operator, and W_1, W_2, b_1, b_2 are learnable parameters of the task encoder with parameter set ϕ. The stop-gradient operation $\text{sg}(\cdot)$ blocks gradients only along the routing path, while the language branch remains trainable through the policy objectives. Each row of $r_\phi(p) \in \mathbb{R}^{L_S \times K}$ provides a task-dependent residual for the K local candidate slots of one student layer. The task instruction specifies the manipulation but not the position of the current noisy action along the flow trajectory. We therefore complement the task-dependent residual with flow-stage conditioning.

4.3 Conditional Cross-Layer Attention Distillation

The task-flow router determines how much each nearby teacher layer should contribute, but these routing weights must still be converted into a training target for the student. We therefore use them

to combine the teacher attention maps associated with each retained student layer and distill the resulting conditional target to the student, providing intermediate supervision adapted to each task.

For each teacher or student layer, we extract the attention submatrix whose rows correspond to action-query tokens and whose columns correspond to vision-language prefix tokens. We average this submatrix over attention heads to obtain $A_j^T \in \mathbb{R}^{N_q \times N_p}$ for teacher layer j and $A_l^S \in \mathbb{R}^{N_q \times N_p}$ for student layer l, where N_q and N_p denote the numbers of action-query tokens and vision-language prefix tokens, respectively. The sample dimension is omitted for clarity. We then construct the teacher target for student layer l as a weighted combination of nearby teacher attention maps:

$$M_l^T(p,t) = \sum_{\rho \in \mathcal{Q}_l} \alpha_{l,\rho}(p,t) A^T_{j_{l,\rho}}, \tag{6}$$

where $j_{l,\rho}$ indexes the teacher layer corresponding to candidate offset ρ, and $\alpha_{l,\rho}(p,t)$ is the routing weight defined in the previous section. The router changes only the source of supervision during training. It does not modify the student architecture or determine which student layers are executed. Thus, the same student layer can receive different weighted teacher targets across task instructions and flow stages while following the same fixed computation graph at inference. Before comparison, we normalize both the teacher and student attention maps over the vision-language prefix dimension using the row-wise normalization operator $\mathcal{N}_P(\cdot)$ to obtain valid attention probability distributions:

$$Q_l^T = \mathcal{N}_P(M_l^T), \;\; Q_l^S = \mathcal{N}_P(A_l^S). \tag{7}$$

Formally, the conditional attention-distillation objective is then defined as

$$\mathcal{L}_{\text{attn}} = \frac{1}{L_S} \sum_{l=0}^{L_S-1} \mathbb{E}_{n,q}\left[D_{\text{KL}}\left(Q_l^T[n,q] \| Q_l^S[n,q] \right) \right], \tag{8}$$

where n indexes training samples, q indexes action queries, and $Q_l^T[n,q]$ denotes the q-th row of the normalized attention map for sample n. The KL divergence is computed over vision-language prefix keys and averaged over queries, samples, and retained student layers.

Because the dense teacher is frozen, this objective updates the student attention and routing weights rather than the teacher representations. The router is therefore learned through the same attention-transfer objective whose target it constructs, without requiring separate teacher-layer labels. Soft mixing also allows several nearby teacher layers to contribute when useful instead of forcing a discrete layer-selection decision. Having defined the conditional intermediate supervision, we next combine it with the original flow objective, teacher velocity distillation, and a structural regularizer to train the complete model through joint optimization of student and router.

4.4 Overall Objective

The previous modules determine where the intermediate teacher supervision comes from and how it is transferred to the student. We now combine this conditional attention supervision with the original policy objectives to train the student and router jointly. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_1 \mathcal{L}_{\text{vel}} + \lambda_2 \mathcal{L}_{\text{attn}} + \lambda_3 \mathcal{L}_{\text{mono}}, \tag{9}$$

where λ_1, λ_2, and λ_3 are hyperparameters that weight the velocity-distillation, attention-distillation, and monotonicity losses, respectively. The first two terms preserve action-level supervision:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}[\|v_t - u_t\|_2^2], \;\; \mathcal{L}_{\text{vel}} = \mathbb{E}[\|v_t - v_t^T\|_2^2], \tag{10}$$

where $v_t = F_\theta(\mathbf{o}, x_t, t)$ is the student's predicted velocity, $v_t^T = F_{\theta_T}(\mathbf{o}, x_t, t)$ is the teacher's predicted velocity, and $u_t = \epsilon - a$ is the target velocity defined in Section 3.

The flow loss trains the student from demonstrated actions, whereas velocity distillation encourages it to follow the dense teacher's predicted action field. These two objectives act directly on the policy and do not train the router. In contrast, $\mathcal{L}_{\text{attn}}$ provides both intermediate representation supervision to the student and the main data-dependent signal for learning the task-flow routing weights.

Although the local candidate sets prevent large cross-depth jumps, neighboring student layers are routed independently and could still reverse their expected teacher-layer order. To preserve the coarse depth progression inherited from the dense teacher without enforcing a fixed alignment, we

introduce a weak ordering regularizer. We define the expected teacher-layer index for student layer l as a weighted average over its valid local teacher candidates:

$$\mu_l(p,t) = \sum_{\rho \in \mathcal{Q}_l} \alpha_{l,\rho}(p,t) j_{l,\rho}, \tag{11}$$

and penalize inversions between consecutive student layers:

$$\mathcal{L}_{\text{mono}} = \frac{1}{L_S - 1} \sum_{l=0}^{L_S-2} \max(0, \mu_l - \mu_{l+1}). \tag{12}$$

This regularizer does not force a fixed teacher-student correspondence across different task instructions and flow stages. It only discourages reversals in the expected teacher-layer order, leaving the router free to shift its local preference according to task instructions and flow stages.

During optimization, the student receives gradients from $\mathcal{L}_{\text{flow}}$, $\mathcal{L}_{\text{vel}}$, and $\mathcal{L}_{\text{attn}}$, whereas the router is updated through $\mathcal{L}_{\text{attn}}$ and $\mathcal{L}_{\text{mono}}$. The dense teacher remains frozen, and the teacher and student use the same demonstration action a, sampled noise ϵ, and flow time t for each training sample. After distillation, we remove the teacher and all routing components, leaving only the fixed depth-compressed student for deployment. The complete training procedure is summarized in Appendix A.

5 Experiments

5.1 Experimental Setup

Baselines. The baseline is a matched L8 reproduction of Shallow-π (Jeon et al., 2026), following its fixed middle-layer attention-distillation design under the same teacher, student initialization, data, training budget, and evaluation protocol as `TFC-KD`. The routing ablation further includes Static Soft Mix, Task-only, Time-only, a parameter-matched Generic Adaptive Router, and Task+Time without $\mathcal{L}_{\text{mono}}$. For π_0, we compare the dense model, matched Shallow-π L8, and `TFC-KD` L8.

Configurations. We evaluate on four 10-task LIBERO suites (Liu et al., 2023). Primary $\pi_{0.5}$ models are trained on the full 40-task mixture for 30K optimizer steps with batch size 64. For the main compression experiments, we reduce the teacher's $L_T = 18$ paired Transformer layers to either $L_S = 8$ or $L_S = 6$ student layers. Following the rounding formula in Section 4.1, the corresponding depth-aligned indices are $\mathbf{c}_8 = [0, 2, 5, 7, 10, 12, 15, 17]$ and $\mathbf{c}_6 = [0, 3, 7, 10, 14, 17]$, respectively. For local teacher-layer supervision, we use a neighborhood radius of $r = 2$, so each student layer considers at most $K = 2r+1 = 5$ teacher-layer candidates with relative offsets $\{-2, -1, 0, +1, +2\}$ around its depth-aligned teacher layer. Candidates falling outside the valid teacher-layer range are masked out. Evaluation uses environment seeds 7, 17, and 27 with 50 episodes per task, yielding 6,000 closed-loop episodes per checkpoint. We also evaluate the L8 compression protocol on π_0, zero-shot perturbation transfer on LIBERO-Plus, and a matched CALVIN-5 balanced atomic subset.

Training Cost. The Task-Flow router introduces only 0.134M additional trainable parameters, corresponding to 0.00587% of the L8 student. Under the matched four-A800 L8 training configuration, steady-state training time increases from 4.422 to 4.609 s/step, corresponding to a 4.24% overhead. Because all routing components are removed after distillation, this additional cost is incurred only during training and does not affect deployment efficiency. Detailed matched training configuration and router parameterization are provided in Appendix B.

5.2 Main Results and Deployment Efficiency

Table 1 jointly summarizes task performance and deployment efficiency across backbones and compression depths. LIBERO reports average success over the four task suites. LIBERO-Plus reports zero-shot perturbation performance using the corresponding LIBERO-trained checkpoint, without perturbation-specific training or checkpoint selection. Deployment latency is measured on one A800 with batch size 1, 10 denoising steps, and the same input configuration.

Matched-Depth Improvement. At the same eight-layer student depth, `TFC-KD` improves the matched Shallow-π baseline from 95.15% to 96.73% on LIBERO and from 75.85% to 77.72% on LIBERO-Plus. Both L8 students have the same 2.280B deployed parameters and 1.761T FLOPs, and their measured A800 latencies are similar (109.99 ms versus 108.32 ms). The improvement therefore does not require a larger or more computationally expensive deployed student.

Table 1: Main task performance and deployment efficiency. Colored values indicate changes relative to the corresponding dense backbone: success-rate differences are reported in percentage points, while parameter, FLOP, and latency reductions are reported in relative percentages. Red denotes a favourable change and blue denotes a decrease in task success.

Backbone	Model	LIBERO	LIBERO-Plus	Params	FLOPs	Latency
$\pi_{0.5}$ (Black et al., 2025a)	Dense	96.28%	75.21%	3.617B	3.413T	206.41 ms
	Shallow-π L8	95.15% (-1.13)	75.85% (+0.64)	2.280B (↓36.96%)	1.761T (↓48.40%)	109.99 ms (↓46.71%)
	TFC-KD L8	96.73% (+0.45)	77.72% (+2.51)	2.280B (↓36.96%)	1.761T (↓48.40%)	108.32 ms (↓47.52%)
	Shallow-π L6	94.18% (-2.10)	73.18% (-2.03)	2.012B (↓44.37%)	1.431T (↓58.07%)	89.90 ms (↓56.45%)
	TFC-KD L6	95.52% (-0.76)	75.46% (+0.25)	2.012B (↓44.37%)	1.431T (↓58.07%)	89.75 ms (↓56.52%)
π_0 (Black et al., 2025b)	Dense	93.87%	61.41%	3.501B	2.785T	175.96 ms
	Shallow-π L8	92.28% (-1.59)	61.63% (+0.22)	2.227B (↓36.39%)	1.483T (↓46.75%)	96.56 ms (↓45.12%)
	TFC-KD L8	93.42% (-0.45)	65.54% (+4.13)	2.227B (↓36.39%)	1.483T (↓46.75%)	98.87 ms (↓43.81%)

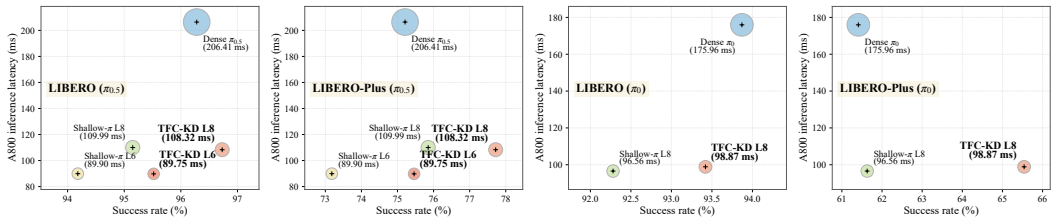

Figure 2: **Success–latency trade-off across LIBERO and LIBERO-Plus for $\pi_{0.5}$ and π_0.** Each point represents one deployed policy according to its task success rate and measured A800 inference latency. The diameter of each point is proportional to the measured latency on the y-axis.

Stronger Compression. The improvement persists under stronger six-layer compression. TFC-KD increases LIBERO success from 94.18% to 95.52% and LIBERO-Plus success from 73.18% to 75.46% relative to the matched Shallow-π L6 baseline. On LIBERO, the L6 student remains within 0.76 percentage points of the dense $\pi_{0.5}$ model (96.28% $\rightarrow$ 95.52%) while reducing deployed parameters from 3.617B to 2.012B and FLOPs from 3.413T to 1.431T. Its measured A800 latency decreases from 206.41 ms to 89.75 ms, corresponding to a 2.30$\times$ speedup. We report full suite-wise and statistical results for the L6 study in Appendix G.2.

Cross-Backbone Transfer. The same pattern also transfers to π_0. At L8, TFC-KD improves the matched Shallow-π baseline from 92.28% to 93.42% on LIBERO and from 61.63% to 65.54% on LIBERO-Plus. The two L8 students have identical deployed parameter counts and FLOPs (2.227B and 1.483T). Their small latency difference, 96.56 ms versus 98.87 ms, is treated as profiling variation rather than a change in inference complexity. Full LIBERO and LIBERO-Plus breakdowns, together with an additional policy-noise repeatability check, are reported in Appendix G.1. Figure 2 further visualises the success-rate–latency trade-off across both backbones and evaluation settings, providing a direct comparison of task performance and deployment efficiency. At matched student depth, TFC-KD consistently achieves higher task success than Shallow-π while retaining comparable measured inference latency.

5.3 Ablation Study

We evaluate the contribution of each component in TFC-KD through controlled ablation experiments on LIBERO. We systematically remove or modify individual components to isolate their effects.

Ablation Study. To isolate the effect of each design choice, we construct the following variants under the same $\pi_{0.5}$ L8 backbone and training protocol. Shallow-π serves as the matched fixed-layer distillation baseline. (1) Static Soft Mix uses the same local teacher-layer candidates but removes task and flow conditioning. (2) Task-only retains task-conditioned routing but removes flow-stage conditioning. (3) Time-only retains flow-stage routing but removes task conditioning. (4) Generic Router uses a parameter-matched sample-adaptive router without the explicit Task-Flow decomposition. (5) Task+Time

Table 2: Task-flow routing ablation study with L8 VLA students.

Variant	Avg. SR (%)
Shallow-π	95.15
Static Soft Mix	93.47
Task-only	95.28
Time-only	95.44
Generic Router	95.48
Task+Time w/o $\mathcal{L}_{\text{mono}}$	96.27
TFC-KD	**96.73**

w/o $\mathcal{L}_{\text{mono}}$ retains the complete Task-Flow routing mechanism but removes the ordering regularizer. The full TFC-KD includes both Task-Flow conditioning and $\mathcal{L}_{\text{mono}}$.

As shown in Table 2, Static Soft Mix decreases the average success rate from 95.15% to 93.47%, indicating that simply mixing multiple teacher layers is insufficient. Task-only and Time-only reach 95.28% and 95.44%, respectively, while the parameter-matched Generic Router reaches 95.48%. Explicit Task-Flow conditioning further improves the success rate to 96.27%, and adding the ordering regularizer increases it to 96.73%. These results show that the gain comes from the structured use of task-instruction and flow-stage information rather than from adaptive teacher-layer mixing alone. Table 2 reports the training-seed-42 averages for compactness; the complete suite-wise breakdown and robustness over training seeds 42, 52, and 62 are reported in Appendix E.

5.4 Zero-Shot Perturbation Robustness on LIBERO-Plus

We evaluate all 10,030 LIBERO-Plus perturbation variants (Fei et al., 2026) under environment seeds 7, 17, and 27, yielding 30,090 closed-loop rollouts per model. No perturbation data or LIBERO-Plus results are used for training, checkpoint selection, or model tuning. We report mean $\pm$ standard deviation across the three evaluation environment seeds. The full evaluation and aggregation protocols are provided in Appendix C.1.

On $\pi_{0.5}$ L8, TFC-KD reaches 77.72$\pm$0.33% overall success, compared with 75.85$\pm$0.66% for the matched Shallow-π baseline. The improvement also transfers to π_0 L8, where overall success increases from 61.63$\pm$0.75% to 65.54$\pm$0.68%.

Under stronger L6 compression on $\pi_{0.5}$, TFC-KD improves overall LIBERO-Plus success from 73.18% to 75.46%. At the suite level, gains are observed on Spatial, Object, and Goal, contributing to the overall improvement. For the primary $\pi_{0.5}$ L8 setting, full suite-wise and perturbation-category results are reported in Appendix F. The corresponding π_0 L8 and $\pi_{0.5}$ L6 results are reported in Appendices G.1 and G.2, respectively.

5.5 Cross-Benchmark Validation on CALVIN-5

We further evaluate on a matched CALVIN-5 balanced atomic subset consisting of five independently reset manipulation tasks. Dense $\pi_{0.5}$, Shallow-π L8, and TFC-KD L8 are trained for 10K steps. Evaluation uses environment seeds 7, 17, and 27 with 50 episodes per task and seed, yielding 750 episodes per model for each maximum-step setting. The complete evaluation protocol is provided in Appendix C.2.

As shown in Table 3, TFC-KD L8 improves over Shallow-π L8 by 6.40, 11.73, and 12.67 percentage points when the maximum episode length is 240, 160, and 120 steps, respectively.

The performance gap between TFC-KD and Shallow-π increases as the maximum number of allowed steps is reduced, indicating that the advantage of TFC-KD persists under more restrictive execution limits.

Table 3: CALVIN-5 results on the matched atomic subset.

Model	SR@240	SR@160	SR@120	Steps@240
Dense $\pi_{0.5}$	100.00	100.00	99.73	56.98 / 55
Shallow-π L8	93.47	86.13	84.13	80.06 / 67
TFC-KD L8	**99.87**	**97.87**	**96.80**	**68.38 / 64**
Improve $\uparrow$	+6.40%	+11.73%	+12.67%	-11.68 step

SR is reported in percent. Steps@240 reports mean / median successful-episode completion steps. The last row reports success-rate differences in percentage points and the difference in mean completion steps. Bold indicates the better result between the two L8 students; Dense $\pi_{0.5}$ is the uncompressed reference.

At the 240-step setting, successful episodes from TFC-KD require 68.38 interaction steps on average, compared with 80.06 for Shallow-π L8. This corresponds to 11.68 fewer steps on average, or a 14.6% reduction in the number of interactions required for successful task completion. Full per-seed, per-task, completion-step, and paired statistical results are reported in Appendix I.

6 Mechanism Analysis

We use two complementary post-training diagnostics to examine whether the learned routing behavior is consistent with the proposed Task-Flow conditional supervision mechanism.

Router–Gradient Agreement. For each probe, we independently identify the teacher layer whose Attention-KD gradient has the highest cosine compatibility with the action-objective gradient, and

refer to it as the gradient-preferred teacher layer. This diagnostic is computed only after training and is never used as supervision for the router.

Across five probe seeds and 40 LIBERO tasks, the learned router selects the gradient-preferred teacher layer in 46.58% of the probes, compared with a candidate-count-corrected chance rate of 23.33%. This above-chance agreement provides evidence of a correspondence between the learned teacher-layer preference and the independent gradient-based diagnostic. Full diagnostic definitions and statistical details are provided in Appendix H.

Action-Gradient Compatibility. We further examine whether the mixed Attention-KD update is directly compatible with the action objective. As shown in Table 4, the full `TFC-KD` achieves the highest mean action-gradient cosine of 0.04153. Its advantage over Time-only and Static Soft Mix has 95% confidence intervals entirely above zero, while Task-only remains numerically close to the full model on this diagnostic. Together with the task- and flow-dependent variation in preferred teacher layers reported in Appendix H, these results provide complementary first-order evidence consistent with the proposed Task-Flow conditional supervision mechanism.

Table 4: Action-gradient compatibility.

Routing	Mean cosine	Gain	95% CI
`TFC-KD`	**0.04153**	–	–
Task-only	0.04076	+0.00076	$[-0.00030, 0.00180]$
Time-only	0.03681	+0.00472	$[0.00068, 0.00863]$
Static Soft Mix	0.03656	+0.00497	$[0.00095, 0.00890]$

Gain denotes `TFC-KD` minus the corresponding baseline; CIs are paired-bootstrap 95% intervals.

Layer-Wise Implication. Beyond the overall Task-Flow variation, the gradient diagnostic reveals substantial heterogeneity across student depth. Across the eight student layers, the switch rate of the gradient-preferred teacher layer ranges from 14% to 58%, task disagreement ranges from 19% to 47%, and flow-stage disagreement ranges from 22% to 75% (Appendix H). This pronounced layer-wise variation suggests that the dependence of teacher-layer preference on task instruction and flow stage is itself heterogeneous across student depth: some student layers remain relatively stable, whereas others exhibit substantially stronger conditional variation.

7 Conclusion

In this paper, we study the problem of capability recovery for depth-compressed flow-based VLA models and propose `TFC-KD`, a Task-Flow conditional cross-layer distillation framework. The key idea of `TFC-KD` is to adapt teacher-layer supervision during training according to task instructions and flow stages, rather than assigning each student layer a fixed teacher layer. Specifically, `TFC-KD` first constructs a local set of adjacent teacher-layer candidates for each retained student layer. It then uses Task-Flow conditional routing to combine their attention targets and introduces a monotonicity regularizer to preserve the ordering of teacher supervision across student layers. All routing components are removed after distillation, leaving a fixed depth-compressed policy with no additional routing computation at inference. Experiments on LIBERO, zero-shot LIBERO-Plus perturbations, π_0, stronger L6 compression, and the matched CALVIN-5 balanced atomic subset demonstrate consistent improvements over matched depth-compressed baselines. Controlled ablations and post-training gradient diagnostics further provide evidence consistent with the proposed conditional supervision mechanism.

8 Limitations and Future Work

The routing design uses a fixed local teacher-layer neighborhood and a predefined flow-stage discretization. Although the layer-wise diagnostic reveals heterogeneous teacher-layer preferences across student depth, all student layers use the same candidate-range rule. This restriction may exclude useful supervision from more distant teacher layers. The current experiments do not systematically characterize sensitivity to candidate width, flow-bin count, routing temperature, or regularization strength. Moreover, training requires a dense teacher forward pass and attention extraction, even though the teacher and routing modules are removed at deployment. Future work will investigate layer-dependent candidate neighborhoods and continuous flow-time conditioning to better accommodate variation in teacher-layer preferences. We also plan to evaluate sensitivity to the routing hyperparameters and study strategies for reducing teacher computation and attention-storage costs during distillation. These extensions will retain the fixed student inference architecture while improving the flexibility and training efficiency of conditional supervision.

AI USE STATEMENT

Generative AI tools were used only for language polishing. All technical content, analyses, experiments, and conclusions were produced and verified by the authors.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we document the implementation and experimental specifications of `TFC-KD` throughout the main paper and appendices. Section 4 specifies the student–teacher layer mapping, local teacher candidates, Task-Flow router, training objectives, loss weights, and gradient paths. Section 5 describes the training and evaluation configurations, seeds, A800 profiling, and matched CALVIN-5 cross-benchmark protocol. Appendices A–I provide implementation and training details, LIBERO-Plus and CALVIN-5 evaluation protocols, related work, training pseudocode, full ablation and training-seed robustness results, LIBERO-Plus perturbation breakdowns, cross-backbone and compression-depth results, gradient-mechanism diagnostic details, and matched CALVIN-5 balanced atomic results.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. doi: 10.15607/RSS.2023.XIX.025.

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7028–7036, 2021a. doi: 10.1609/aaai.v35i8.16865.

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5008–5017, 2021b.

Xiao Chen, Changyi Ma, Wenqi Fan, Zhaoxiang Zhang, and Li Qing. C2KD: Cross-layer and cross-head knowledge distillation for small language model-based recommendation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17827–17838, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.917.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. doi: 10.15607/RSS.2023.XIX.026.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189.

Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Heming Cui, Zhizheng Zhang, and He Wang. GraspVLA: a grasping foundation model pre-trained on billion-scale synthetic action data. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1004–1029. PMLR, 2025.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023.

Gongfan Fang, Kunjun Li, Xinyin Ma, and Xinchao Wang. Tinyfusion: Diffusion transformers learned shallow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18144–18154, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, 2026.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.090.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. doi: 10.1007/s11263-021-01453-z.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 7436–7456, 2022. doi: 10.1109/TPAMI.2021.3117837.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 991–1002. PMLR, 2022.

Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-π: Knowledge distillation for flow-based vlas. *arXiv preprint arXiv:2601.20262*, 2026.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, 2024.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20852–20867. PMLR, 2023.

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.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.071.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-flow models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3736–3761. PMLR, 2025.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Jie Song, Ying Chen, Jingwen Ye, and Mingli Song. Spot-adaptive knowledge distillation. *IEEE Transactions on Image Processing*, 31:3359–3370, 2022. doi: 10.1109/TIP.2022.3170728.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4323–4332, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1441.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788. Curran Associates, Inc., 2020.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pp. 409–424. Springer, 2018.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826. IEEE, 2018. doi: 10.1109/CVPR.2018.00919.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. doi: 10.15607/RSS.2023.XIX.016.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

A TRAINING PROCEDURE OF TFC-KD

In this section, we describe the training procedure of TFC-KD, as summarized in Algorithm 1. Given a dense teacher with L_T Transformer layers, our goal is to train a shallow student with L_S layers while preserving a fixed inference architecture. We first initialize each student layer from a depth-aligned teacher layer according to Section 4.1. For each student layer l, we then construct a local candidate set around its initialization index c_l. Here, K denotes the maximum candidate width, chosen to be odd, and $\rho \in \{0, \dots, K-1\}$ denotes a candidate-slot index. The corresponding teacher-layer index is $j_{l,\rho} = c_l + \rho - \lfloor K/2 \rfloor$. Only slots whose teacher-layer indices fall within $\{0, \dots, L_T - 1\}$ are retained in the valid set $\mathcal{Q}_l$.

Given a training batch (I, p, s, a), where I, p, s, and a denote the visual observations, task instruction, robot state, and target action sequence, respectively. We sample a flow time t and Gaussian noise ϵ to construct the interpolated action x_t and target velocity u_t. The teacher and student process the same inputs to produce velocity predictions and attention maps. We obtain the task representation e_p by masked mean pooling the student language-token features $\{h_k^{\text{lang}}\}$ using the valid-token mask $\{m_k\}$, following the definition in the main text. The stop-gradient operation produces $\widetilde{e}_p$ before it is passed to the task router, preventing routing gradients from propagating through the task-embedding extraction path.

For each student layer, the task-dependent logits $r_\phi(p)$ are combined with the base logits $\beta_{b(t)}$ associated with the current flow stage. A temperature-scaled softmax over the valid candidate slots yields the routing weights $\alpha_{l,\rho}$. These weights combine the head-averaged teacher attention maps into a conditional supervision target, which is normalized using $\mathcal{N}_P$ as defined in the main text. The same weights also determine the expected teacher-layer index μ_l used by the monotonicity regularizer. The student and routing parameters are jointly optimized using the flow-matching, velocity-distillation, attention-distillation, and monotonicity losses. After training, the teacher and all routing components are discarded, and only the fixed shallow student is retained for inference.

B IMPLEMENTATION AND TRAINING DETAILS

Matched Training Configuration. For the primary LIBERO experiments, all $\pi_{0.5}$ models are trained on the same 40-task mixture for 30K optimizer steps with global batch size 64. The matched Shallow-π L8 baseline and TFC-KD L8 use the same frozen dense teacher, student initialization, training data, optimization budget, and evaluation protocol. The only difference is the source of intermediate teacher supervision during training. The same matched-comparison principle is used for the π_0 L8 and $\pi_{0.5}$ L6 experiments.

Router Parameter Count. The Task-Flow router contains 120 learnable flow-stage base logits, a $2048 \to 64$ task-conditioning layer with 131,136 parameters, and a $64 \to 40$ output layer with 2,600 parameters. This gives 133,856 additional trainable parameters in total (0.134M), which corresponds to 0.00587% of the 2.280B L8 student.

C ADDITIONAL EXPERIMENTAL PROTOCOLS

C.1 LIBERO-PLUS EVALUATION PROTOCOL

All models evaluated on LIBERO-Plus are trained only on the original LIBERO data. No LIBERO-Plus perturbation data or evaluation results are used for training or checkpoint selection. Each model is evaluated using its corresponding LIBERO-trained checkpoint.

The complete benchmark contains 10,030 perturbation variants per environment seed: 2,402 Spatial, 2,518 Object, 2,591 Goal, and 2,519 Long. We evaluate environment seeds 7, 17, and 27, with one rollout per variant, yielding 30,090 rollouts per model.

Observations consist of third-person RGB and wrist RGB images under the two-image input protocol. Images are rendered at 256×256, rotated by 180 degrees, and resized with padding to 224×224. The action horizon is 10, and the first five predicted actions are executed before replanning. We wait 10 environment steps before policy execution. Maximum episode lengths are 220, 280, 300, and 520 for Spatial, Object, Goal, and Long, respectively. Videos are disabled during the full evaluation, and binary episode-success records are stored.

Aggregation. For each environment seed, Overall is computed as the pooled episode success rate over all 10,030 perturbation variants, so each perturbation instance receives equal weight. The re-

Algorithm 1 Training procedure of `TFC-KD`.

Require: Teacher F_{θ_T} with L_T layers; student F_θ with L_S layers; odd candidate width K; task router r_ϕ; flow-stage logits β; temperature schedule τ; loss weights $\lambda_1, \lambda_2, \lambda_3$
Ensure: Trained shallow student F_θ

for $l = 0, \ldots, L_S - 1$ **do**
  $c_l \leftarrow \left\lfloor \dfrac{l(L_T - 1)}{L_S - 1} + 0.5 \right\rfloor$
  $j_{l,\rho} \leftarrow c_l + \rho - \lfloor K/2 \rfloor, \quad \rho = 0, \ldots, K - 1$
  $\mathcal{Q}_l \leftarrow \{\rho \in \{0, \ldots, K-1\} : 0 \le j_{l,\rho} < L_T\}$
  Initialize student layer S_l from teacher layer T_{c_l}
end for
Freeze F_{θ_T}
for each training batch (I, p, s, a) **do**
  Sample $t \sim \mathcal{U}(0,1)$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $x_t \leftarrow (1-t)a + t\epsilon; \quad u_t \leftarrow \epsilon - a$
  $(\hat{v}_t, \{A_l^S\}, \{h_k^{\text{lang}}\}) \leftarrow F_\theta(I, p, s, x_t, t)$
  $(\hat{v}_t^T, \{A_j^T\}) \leftarrow F_{\theta_T}(I, p, s, x_t, t)$ without gradients
  $e_p \leftarrow \dfrac{\sum_k m_k h_k^{\text{lang}}}{\sum_k m_k}; \quad \widetilde{e}_p \leftarrow \text{sg}(e_p)$
  $r_\phi(p) \leftarrow W_2 \,\text{SiLU}\,(W_1 \widetilde{e}_p + b_1) + b_2$
  $b \leftarrow b(t)$; set τ according to the training schedule
  for $l = 0, \ldots, L_S - 1$ **do**
    $\alpha_l \leftarrow \text{Softmax}_{\mathcal{Q}_l}\big((\beta_{b,l} + r_{\phi,l}(p))/\tau\big)$
    $M_l^T \leftarrow \sum_{\rho \in \mathcal{Q}_l} \alpha_{l,\rho} \,\text{HeadAvg}(A_{j_{l,\rho}}^T)$
    $Q_l^T \leftarrow \mathcal{N}_P(M_l^T); \quad Q_l^S \leftarrow \mathcal{N}_P(\text{HeadAvg}(A_l^S))$
    $\mu_l \leftarrow \sum_{\rho \in \mathcal{Q}_l} \alpha_{l,\rho} j_{l,\rho}$
  end for
  Compute $\mathcal{L}_{\text{flow}}$ and $\mathcal{L}_{\text{vel}}$ from $\hat{v}_t$, u_t, and $\hat{v}_t^T$
  Compute $\mathcal{L}_{\text{attn}}$ from $\{Q_l^T, Q_l^S\}$ and $\mathcal{L}_{\text{mono}}$ from $\{\mu_l\}$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{flow}} + \lambda_1 \mathcal{L}_{\text{vel}} + \lambda_2 \mathcal{L}_{\text{attn}} + \lambda_3 \mathcal{L}_{\text{mono}}$
  Back-propagate $\mathcal{L}$ and update F_θ, ϕ, and β
end for
Discard F_{θ_T}, r_ϕ, and β
return F_θ

ported mean and sample standard deviation are computed across the three seed-level pooled success rates. A four-suite macro average yields the same overall conclusion but is not used as the primary Overall because the four suites contain slightly different numbers of perturbation variants.

C.2 CALVIN-5 Balanced Atomic Evaluation Protocol

We construct a matched CALVIN-5 balanced atomic subset containing five manipulation tasks: open drawer, move slider left, lift red block from table, turn on lightbulb, and push into drawer. Each task is evaluated independently with a reset environment.

Dense $\pi_{0.5}$, Shallow-π L8, and `TFC-KD` L8 are each trained for 10,000 optimization steps. For each maximum-step setting, we evaluate environment seeds 7, 17, and 27 with 50 episodes per task and seed. This gives 250 episodes per model and environment seed, and 750 episodes per model for each maximum-step setting.

The action horizon is 10, and the first five predicted actions are executed before replanning. Initial states are sampled by seed from the pool of first-step states of 20,000 official task sequences. All models use the same tasks, episode identifiers, initial-state sampling, replanning schedule, and success predicates. Environment seeds 7, 17, and 27 are used for evaluation-state sampling.

D Detailed Related Work

This appendix expands the related-work discussion in Section 2, covering generalist robot policies, efficient VLA inference, and adaptive cross-layer distillation.

Generalist and Generative Robot Policies. Language-conditioned robot learning includes multi-task imitation methods such as BC-Z (Jang et al., 2022) and large generalist policies such as RT-1,

RT-2, PaLM-E, RoboFlamingo, Octo, and OpenVLA (Brohan et al., 2023; Zitkovich et al., 2023; Driess et al., 2023; Li et al., 2024; Ghosh et al., 2024; Kim et al., 2025). Generative action models use action chunking, diffusion, and flow matching for continuous control (Zhao et al., 2023; Ho et al., 2020; Chi et al., 2023; Liu et al., 2025; Lipman et al., 2023). π_0 and $\pi_{0.5}$ combine pretrained vision-language representations with flow-based action generation (Black et al., 2025b;a). These models repeatedly evaluate deep Transformer stacks during action generation, making inference efficiency an important deployment issue.

Efficient VLA Inference and Compression. Several methods reduce robot-policy deployment costs. Efficient attention and input-dependent computation have been studied broadly in deep networks (Dao et al., 2022; Wang et al., 2018; Wu et al., 2018; Han et al., 2022), and depth reduction has been explored for diffusion Transformers (Fang et al., 2025). In robotics, Consistency Policy reduces iterative diffusion-control cost through consistency distillation (Prasad et al., 2024). DeeR-VLA varies the active model depth during inference (Yue et al., 2024), while FLOWER improves VLA efficiency through architectural design and model scaling (Reuss et al., 2025). Structural depth compression provides a fixed, smaller computation graph. Shallow-π (Jeon et al., 2026) is the closest work in this setting. It physically reduces Transformer depth in flow-based VLAs and applies action-to-vision-language attention distillation at a fixed middle Transformer layer. We adapt this design to a matched L8 reproduction under our unified pipeline. Its results show that intermediate attention supervision helps a shallow student recover performance after depth reduction. The teacher layer used for this supervision remains fixed. We study whether teacher-layer selection for supervision should depend on the current task instruction and flow stage.

Knowledge Distillation and Adaptive Supervision. Knowledge distillation transfers information from a stronger teacher to a compact student (Hinton et al., 2015; Gou et al., 2021). Intermediate supervision has been studied through feature hints and attention transfer (Romero et al., 2015; Zagoruyko & Komodakis, 2017). Transformer compression methods such as Patient Knowledge Distillation, DistilBERT, TinyBERT, and MiniLM also use intermediate or representation-level transfer (Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020). Other work studies contrastive distillation and knowledge review (Tian et al., 2020; Chen et al., 2021b). Several methods allow cross-layer supervision to adapt. SemCKD learns soft associations between teacher and student layers (Chen et al., 2021a). SAKD changes distillation locations across samples and training stages (Song et al., 2022). TED introduces task-aware layer-wise transfer (Liang et al., 2023). C2KD dynamically routes cross-layer supervision (Chen et al., 2025). These methods make layer-wise knowledge transfer adaptive through soft layer association, sample- or stage-dependent distillation locations, task-aware filtering, or dynamic cross-layer routing. However, they do not adapt the teacher layer used for supervision to the flow stage of iterative action generation in flow-based VLAs. `TFC-KD` instead uses the student layer, task instruction, and flow stage to determine task- and flow-dependent mixture weights over adjacent teacher layers. The router is removed after distillation, leaving a fixed depth-compressed inference architecture at deployment.

E Full Ablation and Training-Seed Robustness

Table 2 in the main text reports only the average success rates for training seed 42 for compactness. Here we provide the corresponding suite-wise breakdown and additional robustness results over training seeds 42, 52, and 62.

Suite-wise ablation. Table 5 expands the seed-42 ablation to the four LIBERO suites.

The full model improves over the matched Shallow-π baseline on all four LIBERO suites. Static Soft Mix performs better than Shallow-π on Spatial and Object but substantially worse on Goal and Long, resulting in a lower overall average. Task-only and Time-only recover most of this degradation, while explicit Task-Flow conditioning without the ordering regulariser further increases the average success rate to 96.27%. Adding $\mathcal{L}_{\text{mono}}$ yields the full-model result of 96.73%.

Training-seed robustness. We additionally repeat the main routing controls with training seeds 52 and 62. Table 6 reports the resulting average LIBERO success rates together with the seed-42 results used in the main-text ablation. The full `TFC-KD` model improves over Shallow-π by 1.58, 1.52, and 1.61 percentage points for training seeds 42, 52, and 62, respectively, corresponding to a paired gain of $+1.57 \pm 0.05$ percentage points.

Table 5: Full suite-wise ablation on LIBERO for training seed 42.
All variants use the same L8 student depth.

Variant	Mix	Task	Time	Mono	Spatial	Object	Goal	Long	Avg.
Shallow-π	✗	✗	✗	✗	97.47	96.53	95.53	91.07	95.15
Static Soft Mix	✓	✗	✗	✗	98.13	97.73	93.93	84.07	93.47
Task-only	✓	✓	✗	✗	98.33	97.80	95.47	89.53	95.28
Time-only	✓	✗	✓	✗	98.47	97.87	96.13	89.27	95.44
Generic Router	✓	Generic		✗	98.53	97.60	96.07	89.73	95.48
Task+Time w/o $\mathcal{L}_{\text{mono}}$	✓	✓	✓	✗	98.67	97.53	96.87	92.00	96.27
TFC-KD	✓	✓	✓	✓	**98.73**	**98.00**	**97.27**	**92.93**	**96.73**

Table 6: Training-seed robustness of the main L8 ablation controls.
The last row reports mean $\pm$ standard deviation across training seeds.

Training seed	Shallow-π	Generic Router	Task+Time w/o $\mathcal{L}_{\text{mono}}$	TFC-KD
42	95.15	95.48	96.27	**96.73**
52	95.08	95.63	96.17	**96.60**
62	95.22	95.42	96.33	**96.83**
Mean $\pm$ std	95.15$\pm$0.07	95.51$\pm$0.11	96.26$\pm$0.08	**96.72$\pm$0.12**

Table 7: LIBERO-Plus suite-wise results for $\pi_{0.5}$.

Model	Spatial	Object	Goal	Long	Overall
Dense $\pi_{0.5}$	81.96$\pm$1.21	81.24$\pm$0.63	71.56$\pm$0.72	66.51$\pm$0.47	75.21$\pm$0.43
Shallow-π L8	85.64$\pm$1.13	**81.35$\pm$0.50**	66.64$\pm$0.85	70.49$\pm$0.53	75.85$\pm$0.66
TFC-KD L8	**87.95$\pm$0.54**	81.18$\pm$0.18	**71.86$\pm$0.45**	**70.54$\pm$0.70**	**77.72$\pm$0.33**

Table 8: LIBERO-Plus results by perturbation category for $\pi_{0.5}$.

Model	Camera	Robot	Language	Light	Background	Noise	Layout
Dense $\pi_{0.5}$	58.24$\pm$0.67	**66.04$\pm$0.79**	74.39$\pm$0.91	90.98$\pm$0.61	**89.10$\pm$0.23**	74.56$\pm$0.70	82.23$\pm$0.17
Shallow-π L8	54.85$\pm$0.23	63.68$\pm$1.29	**81.59$\pm$1.96**	93.14$\pm$1.47	86.40$\pm$2.50	77.49$\pm$0.57	82.34$\pm$0.62
TFC-KD L8	**59.77$\pm$0.16**	64.90$\pm$1.45	81.54$\pm$0.26	**94.45$\pm$0.81**	89.06$\pm$0.53	**78.49$\pm$0.10**	**84.39$\pm$0.68**

Structured Task-Flow conditioning without $\mathcal{L}_{\text{mono}}$ outperforms the Generic Router by 0.79, 0.54, and 0.91 percentage points across the three seeds, giving a mean paired gain of $+0.75 \pm 0.19$ percentage points. Adding the ordering regulariser provides a further $+0.46 \pm 0.04$ percentage-point gain on average. Accordingly, the ordering

$$\texttt{TFC-KD} > \text{Task+Time w/o } \mathcal{L}_{\text{mono}} > \text{Generic Router}$$

holds for all three training seeds.

F Full LIBERO-Plus Perturbation Results

The detailed evaluation and aggregation protocols are provided in Appendix C.1. Here we report the full suite-wise and perturbation-category results for the primary $\pi_{0.5}$ L8 setting omitted from the main text. For environment seeds 7, 17, and 27, the seed-level pooled Overall success rates are 75.53/75.37/74.73 for Dense $\pi_{0.5}$, 75.80/76.53/75.21 for Shallow-π L8, and 77.68/78.08/77.42 for TFC-KD L8, respectively.

Category denominators across the three environment seeds are 4,797 Camera, 4,650 Robot, 4,611 Language, 3,426 Light, 3,228 Background, 4,803 Noise, and 4,575 Layout, totalling 30,090 rollouts per model. Relative to Shallow-π L8, TFC-KD improves success in six of the seven perturbation categories, while Language remains essentially unchanged.

Table 9: LIBERO suite-wise results for π_0 L8.

Model	Spatial	Object	Goal	Long	Avg.
Dense π_0	96.40	**98.67**	**95.27**	85.13	**93.87**
Shallow-π L8	95.93	94.53	93.47	85.20	92.28
`TFC-KD` L8	**96.67**	96.40	93.53	**87.07**	93.42

Table 10: LIBERO-Plus suite-wise results for π_0 L8 models.

Model	Spatial	Object	Goal	Long	Overall
Dense π_0	68.89±0.51	**74.36±0.44**	44.81±0.71	58.42±1.32	61.41±0.68
Shallow-π L8	**77.17±1.03**	71.42±0.29	38.94±1.30	60.37±0.81	61.63±0.75
`TFC-KD` L8	76.16±0.91	73.63±0.65	**47.82±0.87**	**65.54±0.79**	**65.54±0.68**

Table 11: LIBERO-Plus perturbation results for π_0 L8 models.

Model	Camera	Robot	Language	Light	Background	Noise	Layout	Overall
Dense π_0	56.29±1.29	**18.62±1.13**	**65.34±0.43**	83.07±1.02	74.29±0.62	71.37±0.73	70.56±2.00	61.41±0.68
Shallow-π L8	62.66±0.66	13.81±0.19	64.26±1.06	83.54±0.75	72.83±1.42	72.93±1.93	70.34±0.57	61.63±0.75
`TFC-KD` L8	**65.40±0.92**	16.15±0.39	64.45±1.73	**87.68±0.88**	**80.48±1.56**	**80.55±1.66**	**74.08±0.47**	**65.54±0.68**

Table 12: Policy-noise repeatability for π_0 L8 models.

Model	Exact A	Exact B	Alternate	Exact Agree.	Cross-Noise Agree.
Dense π_0	75.00	75.00	78.57	96.43	75.00
Shallow-π L8	75.00	75.00	79.17	**97.62**	79.17
`TFC-KD` L8	**77.38**	**76.19**	**80.36**	**97.62**	**82.74**

G Cross-Backbone and Compression-Depth Results

G.1 π_0 Cross-Backbone Results

The main text reports the aggregate π_0 result and deployment efficiency. Here we provide the full suite-wise LIBERO results, LIBERO-Plus breakdown, and the controlled policy-noise repeatability study.

Zero-shot LIBERO-Plus. The LIBERO-Plus evaluation follows the same zero-shot protocol described in Appendix C.1.

Across environment seeds 7, 17, and 27, the Overall success rates for Dense/Shallow-π/`TFC-KD` are 61.33/61.86/66.20, 62.13/62.23/65.57, and 60.78/60.80/64.84, respectively. The corresponding `TFC-KD` improvements over Shallow-π L8 are +4.34, +3.34, and +4.04 percentage points.

Relative to Shallow-π L8, `TFC-KD` achieves higher success in all seven perturbation categories, with the largest category-level improvements under Background and Noise. At the suite level, the largest improvements occur on Goal and Long.

Controlled Policy-Noise Repeatability. To examine whether the observed ordering is sensitive to a single policy-action noise realisation, we conduct a small controlled repeatability study with environment seed 7 fixed. Each run contains 168 episodes: six instances for each of seven perturbation categories in each of the four LIBERO suites. Exact A and Exact B use the same explicit policy-noise seed (20270917), whereas Alternate changes only the policy-noise realisation (20270918).

Repeated runs with the same explicit policy-noise realisation achieve 96.43–97.62% episode-level outcome agreement. `TFC-KD` remains above both comparison models in all three runs, including when the policy-noise realisation is changed.

G.2 Stronger L6 Compression

Aggregate task performance and deployment efficiency at L6 are reported in the main text. Here we provide the full suite-wise results and statistical analyses for the stronger six-layer compression

Table 13: LIBERO Results under Stronger L6 Compression. Differences are computed from unrounded success rates; displayed values are rounded to two decimal places.

Model	Spatial	Object	Goal	Long	Avg.
Shallow-π L6	96.13	96.60	95.87	88.13	94.18
`TFC-KD` L6	**98.53**	**97.47**	**97.33**	**88.73**	**95.52**
`TFC-KD` − Shallow-π	+2.40	+0.87	+1.47	+0.60	+1.33

Table 14: LIBERO-Plus results for L6 models.

Model	Spatial	Object	Goal	Long	Overall
Shallow-π L6	81.65	79.71	62.95	**69.09**	73.18
`TFC-KD` L6	**83.54**	**82.86**	**68.91**	67.10	**75.46**
`TFC-KD` − Shallow-π	+1.89	+3.15	+5.96	-1.98	+2.28

setting. Shallow-π L6 and `TFC-KD` L6 use the same dense $\pi_{0.5}$ teacher, six-layer student depth, 18-to-6 initialisation mapping, 30K-step training budget, and evaluation protocol.

Across evaluation seeds 7, 17, and 27, the four-suite averages are 94.55/94.05/93.95 for Shallow-π L6 and 95.50/95.70/95.35 for `TFC-KD` L6, giving 94.18$\pm$0.32 and 95.52$\pm$0.18, respectively. Shallow-π L6 succeeds in 5,651/6,000 episodes and `TFC-KD` L6 in 5,731/6,000. A paired crossed seed-by-base-task bootstrap, stratified by suite with 20,000 resamples, gives a +1.33-percentage-point difference with 95% CI [0.08, 2.50].

The seed-level pooled Overall rates are 73.12/73.95/72.46 for Shallow-π L6 and 75.27/76.10/75.00 for `TFC-KD` L6, corresponding to 73.18$\pm$0.74 and 75.46$\pm$0.57. `TFC-KD` yields 22,706 successful episodes compared with 22,019 for Shallow-π L6. The instance-weighted paired bootstrap gives a +2.28-percentage-point difference with 95% CI [0.06, 4.89], while an equal-suite aggregation gives +2.25 percentage points with 95% CI [1.39, 5.44].

H Gradient-Mechanism Diagnostic Details

The mechanism analysis in the main text is a post-training diagnostic and is not used to train or supervise the Task-Flow router. The mechanism study uses probe seeds $\{7, 17, 27, 37, 47\}$ across 40 LIBERO tasks. For each task, the probe uses two demonstrations and three flow-time bins. Within each bin, continuous flow times are sampled per example; a seeded Gaussian noise realisation is sampled per example and reused across the three bins for that task.

The action-gradient reference is computed from the flow-matching MSE, using mean reduction over batch, action horizon, and action dimensions before differentiation:

$$g_{\text{act}}(p,t) = \nabla_{\Theta_{S-R}} \mathcal{L}_{\text{flow}}(p,t), \quad \mathcal{L}_{\text{flow}} = \text{MSE}(v_\theta(I,p,s,x_t,t), \epsilon - a)\,, \tag{13}$$

where

$$\Theta_{S-R} = \{\theta \in \Theta_S : \texttt{requires_grad}(\theta) = \text{True},\ \theta \notin \Theta_{\text{router}}\}.$$

The dense teacher is frozen, and router parameters are explicitly excluded. This reference contains neither velocity-distillation nor Attention-KD losses, router regularisers, closed-loop success, nor future rollout information.

For each (l, j),

$$g^{\text{KD}}_{l,j} = \nabla_{\Theta_{S-R}} \mathcal{L}^{l,j}_{\text{attn}}$$

is computed in the same global parameter space. Candidate Attention-KD first averages attention heads, renormalises over vision-language keys, sums teacher-to-student KL over keys, and averages over the batch and action-query tokens.

We define the gradient-compatibility score as

$$C_{l,j}(p,t) = \cos\big(g^{\text{KD}}_{l,j}(p,t), g_{\text{act}}(p,t)\big)\,.$$

For each student layer l, the gradient-preferred teacher layer under task p and flow-time condition t:

$$j^*_l(p,t) = \arg\max_{j \in \mathcal{C}_l} C_{l,j}(p,t).$$

Table 15: Layer-wise variation in gradient-preferred teacher layers.

Student layer	Switch rate	Task disagreement	Flow-time disagreement
S_0	14	19	22
S_1	29	23	34
S_2	**58**	**47**	71
S_3	43	37	63
S_4	38	33	**75**
S_5	29	27	51
S_6	43	40	60
S_7	29	28	54
Range	14–58	19–47	22–75

Table 16: Router agreement with gradient diagnostics.

Metric	Value	95% CI
Top-1 agreement	46.58%	[44.50, 48.67]
Chance baseline	23.33%	–
Top-1 advantage	+23.25 pp	[21.21, 25.33]
Spearman(α, C)	0.0746	[0.0490, 0.1010]
Router expected compatibility	0.00826	[0.00592, 0.01071]
Router − uniform	0.00743	[0.00530, 0.00964]
Router − fixed centre	0.00888	[0.00640, 0.01140]

Cosine similarity is accumulated in FP32 across all parameter tensors, which is equivalent to flattening and concatenating the gradients followed by a global ℓ_2 cosine. Unused gradients are treated as zero. For a fixed task, probe mini-batch, and flow-time bin, g_{act} is computed once and reused while all valid (l, j) candidates are scanned. The diagnostic is therefore strictly post-training and supplies no gradient-oracle signal to the learned router.

For each student layer, we first identify a best fixed teacher layer j_l^{fixed} over the probe conditions. The switch rate is defined as

$$\text{SwitchRate}_l = \Pr_{(p,t)} \left[j_l^*(p,t) \neq j_l^{\text{fixed}} \right].$$

It measures how often the gradient-preferred teacher layer under a specific task instruction and flow stage differs from the best single fixed teacher layer for that student layer.

Task disagreement compares the gradient-preferred teacher layer j_l^* across tasks at a fixed flow-time condition, whereas flow-time disagreement compares j_l^* across flow-time bins for a fixed task. Table 15 reports the complete layer-wise statistics omitted from the main text.

All entries in Table 15 are percentages.

For each valid local candidate slot $\rho \in \mathcal{Q}_l$, $\alpha_{l,\rho}$ denotes the learned routing weight, while $C_{l,j_{l,\rho}}$ denotes the corresponding gradient-compatibility score. Top-1 agreement measures whether the teacher layer with the largest routing weight matches the gradient-preferred teacher layer. Spearman(α, C) measures the rank association between routing weights and gradient-compatibility scores across valid teacher-layer candidates. We additionally compare the compatibility induced by the learned routing distribution with uniform candidate weighting and the fixed centre teacher layer.

Confidence intervals are aggregated over 200 seed×task clusters. For the router–gradient correspondence test, candidate labels are permuted within each seed/task/time/layer observation for 2,000 permutations.

The main text reports the mixed Attention-KD cosine comparisons. For completeness, the mean action-gradient cosine of `TFC-KD` is 0.04153 with a cluster-bootstrap 95% CI of [0.02863, 0.05442]. The one-sided paired bootstrap p-values for `TFC-KD` over Time-only and Static Soft Mix are 0.0107 and 0.0082, respectively. The corresponding Top-1 and Spearman candidate-label permutation tests both yield $p < 0.001$. Because the gradient-preferred teacher layer is never exposed to the router during training, these post-training results provide additional evidence of a correspondence between

Table 17: CALVIN-5 successful-episode steps (mean/median).

Max steps	Dense $\pi_{0.5}$	Shallow-π L8	TFC-KD L8
240	56.98 / 55	80.06 / 67	68.38 / 64
160	56.79 / 55	71.36 / 66	66.76 / 64
120	56.75 / 55	68.81 / 66	66.55 / 64

Table 18: CALVIN-5 results by environment seed at 240 steps.

Environment seed	Dense $\pi_{0.5}$	Shallow-π L8	TFC-KD L8
7	**250/250 (100.00%)**	234/250 (93.60%)	**250/250 (100.00%)**
17	**250/250 (100.00%)**	235/250 (94.00%)	**250/250 (100.00%)**
27	**250/250 (100.00%)**	232/250 (92.80%)	249/250 (99.60%)

the learned teacher-layer preference and the independent gradient-based diagnostic. They should be interpreted as local first-order diagnostic evidence rather than as a causal verification of the routing mechanism.

I MATCHED CALVIN-5 BALANCED ATOMIC EVALUATION

The complete evaluation protocol is described in Appendix C.2. This evaluation uses our matched CALVIN-5 balanced atomic subset as a compact cross-benchmark evaluation and should not be interpreted as the official CALVIN long-horizon sequence benchmark. The main text reports aggregate success rates at 240, 160, and 120 maximum steps together with the primary 240-step completion result. Here we provide the full completion-step statistics, per-seed results, paired significance tests, and per-task breakdowns.

Values are reported as mean / median over successful episodes. The 240-step setting is the least affected by truncation and is therefore used as the primary completion-step comparison in the main text. Because these statistics are conditioned on successful episodes, they are reported as descriptive completion-efficiency measures rather than as a paired completion-time comparison over identical successful episodes.

All significance comparisons are paired at the episode level because the compared models use the same tasks, environment seeds, episode identifiers, and initial-state sampling. At 240, 160, and 120 steps, the paired success-rate differences between TFC-KD and Shallow-π L8 are +6.40, +11.73, and +12.67 percentage points, with 95% bootstrap confidence intervals of [4.67, 8.27], [9.33, 14.27], and [10.13, 15.33], respectively. The corresponding "TFC-KD only / Shallow-π only" discordant counts are 49/1, 95/7, and 104/9, with McNemar p-values of 9.06×10^{-14}, 7.85×10^{-21}, and 1.26×10^{-21}.

Table 19: Per-task success rates on CALVIN-5.

Max steps	Model	Open drawer	Move slider left	Lift red block	Turn on lightbulb	Push into drawer
240	Dense $\pi_{0.5}$	100.00	100.00	100.00	100.00	100.00
	Shallow-π L8	100.00	73.33	99.33	100.00	94.67
	TFC-KD L8	100.00	99.33	100.00	100.00	100.00
160	Dense $\pi_{0.5}$	100.00	100.00	100.00	100.00	100.00
	Shallow-π L8	100.00	44.00	98.67	100.00	88.00
	TFC-KD L8	100.00	92.67	100.00	100.00	96.67
120	Dense $\pi_{0.5}$	100.00	100.00	100.00	100.00	98.67
	Shallow-π L8	100.00	41.33	100.00	100.00	79.33
	TFC-KD L8	100.00	90.00	99.33	100.00	94.67

The per-task breakdown shows that the largest differences occur on move-slider-left. At 240, 160, and 120 maximum steps, Shallow-π L8 achieves 73.33%, 44.00%, and 41.33%, whereas TFC-KD L8 achieves 99.33%, 92.67%, and 90.00%, respectively. The difference in push-into-drawer also increases as the maximum number of allowed steps is reduced.