

# CONTRA-KD: Continuous Trajectory Alignment for Knowledge Distillation

Anonymous Authors<sup>1</sup>

## Abstract

While Knowledge Distillation (KD) enables efficient compression of LLMs, current methods focus on matching discrete layer outputs, which fails to capture the continuous, dynamic nature of feature evolution within these models. We propose **CONTRA-KD**, a new paradigm for KD that reframes the process through continuous-time dynamics and the ordinary differential equations (ODE)-based nature of LLMs. We first learn a student-to-teacher correction flow in feature space using rectified flow matching, which provides a geometrically simple and stable transport path between intermediate representations. We then introduce a curriculum-based distillation strategy that guides the student to progressively follow this flow, generating dynamic intermediate targets throughout training. This design enables the student to align not only with the teacher’s representations but also with its underlying feature dynamics, while remaining orthogonal and complementary to standard output-level distillation objectives. Extensive experiments across multiple teacher–student pairs and instruction-following benchmarks demonstrate that CONTRA-KD consistently improves distillation performance and training stability over strong baselines.

## 1. Introduction

Large Language Models (LLMs) have achieved remarkable capabilities, thanks to their immense scale with leading models now contain billions of parameters (Touvron et al., 2023). However, this scale creates a substantial barrier for practical deployment, especially in constrained, low-resource settings. Knowledge Distillation (Hinton et al., 2015) offers a compelling solution, by compressing a large, powerful teacher model into a smaller, faster student model

by transferring knowledge between the two.

The major paradigm in LLMs distillation primarily focuses on transferring knowledge distillation via the final output distributions. The foundation approach (Hinton et al., 2015) minimizes the forward Kullback-Leibler divergence between the teacher’s and student’s logits, encouraging the student to mimic the teacher’s probability distribution. Recent method (Gu et al., 2024) employs reverse KL divergence, forcing the student to approximate the teacher’s distribution more precisely on its own generated samples. Further advancements (Agarwal et al., 2024; Ko et al., 2024) refine this objective by integrating on-policy data generation or adaptive divergence measures to better align the sequence-level probability mass. While being effective, these methods focus on aligning the final output logits, overlooking the rich, dynamic evolution of representations within the model’s intermediate layers. A more profound perspective emerges when we view the Transformer architecture not as a static stack of layers, but as a dynamic system. This interpretation, grounded in the theory of Ordinary Differential Equations (ODEs), treats the layer-wise progression as discrete time steps in a continuous evolution of feature representations (Chen et al., 2019). Within this powerful framework, the teacher’s deep architecture can be understood as a high-resolution solver, tracing a fine-grained path through the feature space. In contrast, the student’s shallower architecture represents a coarser, less accurate solver attempting to follow a similar trajectory.

Recent advancements have moved in this direction. Feature Dynamic Distillation (Gong et al., 2025) aligns not only feature states at discrete layers but also their layer-to-layer changes, providing a more detailed snapshot of the model’s dynamics. However, these methods remain fundamentally tied to a discrete, layer-by-layer view. This often leads to struggles of capturing the global geometric consistency of the teacher’s feature flow. Concurrently, the emerging literature on continuous-time flow matching (Lipman et al., 2023; Liu et al., 2023) provides a powerful lens for modeling probability transport. Instead of relying on stochastic sampling, these methods directly learn continuous velocity fields that connect source and target distributions along straight-line paths. This perspective naturally complements the ODE view of Transformers: distillation can be seen as transporting the student’s hidden representations along trajectories

<sup>1</sup>Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

that align with the teacher’s flow.

Grounded in this insight, we introduce **CONTRA-KD**, a new framework that formulates KD as a rectified flow matching problem in feature space. Rather than forcing the student to replicate the teacher’s full and often noisy dynamics, CONTRA-KD first learns an idealized rectified flow — a straight-line ODE that captures the most direct transformation from any student’s feature state to its corresponding teacher’s state. By learning this flow, not only do we distill the intermediate representation states of the teacher and student model, but we also distill the continuous internal dynamics of teacher to the student’s. In the second stage, the student is trained to align its own layer-wise updates with a discretized simulation of this rectified flow. By doing so, CONTRA-KD replaces brittle, discrete state matching with a continuous dynamical perspective, and substitutes imitation of raw teacher behavior with alignment to an optimal, geometrical trajectory.

The contributions of our study are as follows:

- We formulate knowledge distillation from a continuous-time perspective, viewing both teacher and student models as dynamical systems governed by velocity fields. Under this formulation, effective distillation corresponds to aligning the trajectories induced by their internal dynamics, rather than matching discrete layer-wise representations.
- We propose a new two-stage method of Continuous Trajectory Alignment for Knowledge Distillation (**CONTRA-KD**). In the first stage, CONTRA-KD learns a universal, student-to-teacher correction flow. In the second stage, the student is trained via a discretized simulation of this rectified flow, using it as a dynamic target for more stable and efficient feature evolution.
- We introduce a **curriculum-based trajectory distillation** strategy that progressively increases distillation difficulty as training progresses. By distilling small step dynamics first and gradually extending the step size, this curriculum stabilizes optimization and enables effective velocity and trajectory alignment.

## 2. Backgrounds

### 2.1. Related Work

**Knowledge Distillation of LLMs.** Knowledge Distillation (KD) (Hinton et al., 2015) is a model compression technique where a student learns from a teacher model’s outputs. A line of research has focused on treating distillation as a sequence-level alignment problem (Kim & Rush, 2016). More recent methods have built on this, by using

Kullback–Leibler divergence on student generated outputs (SGOs) (Lin et al., 2020), employing Jensen-Shannon divergence (Agarwal et al., 2024), or Reverse KLD (Gu et al., 2024; Wu et al., 2024).

**Intermediate Representations Distillation.** A complementary line of work explores distilling knowledge from a teacher’s intermediate representations, motivated by the observation that deep models encode rich hierarchical information beyond their outputs (Skean et al., 2025). Early successes in this direction were demonstrated for BERT-style architectures (Sanh et al., 2020; Jiao et al., 2020). More recently, inspired by interpreting Transformers as discretizations of Ordinary Differential Equations (ODEs) (Chen et al., 2019; Lu et al., 2019), feature dynamics distillation methods have framed KD as aligning the trajectories of a coarse ODE solver (the student) with those of a fine-grained solver (the teacher) (Gong et al., 2025). While conceptually appealing, these approaches rely on discrete layer-to-layer differences as proxies for continuous dynamics, which can limit their ability to capture the global geometry of the teacher’s feature flow.

### 2.2. Preliminary

#### 2.2.1. KNOWLEDGE DISTILLATION IN LLM

Knowledge distillation (KD) (Hinton et al., 2015) is a technique for model compression where a large teacher model transfers its knowledge to a smaller student model. The conventional KD framework primarily achieves this by aligning the final output distributions (logits) of the student with those of the teacher. Specifically, for a final-layer student hidden state  $h_S^{L_S}$  and teacher hidden state  $h_T^{L_T}$ , where the student and the teacher have  $L_S$  and  $L_T$  layers respectively, the loss minimizes the Kullback-Leibler (KL) divergence between the probability distributions produced by their respective language modeling heads,  $f_{\text{head}}(\cdot)$ :

$$\mathcal{L}_{\text{KD}} = \mathbb{E}_{x \sim \text{data}} \left[ D_{\text{KL}}(f_{\text{head}}^T(h_T^{L_T}) \| f_{\text{head}}^S(h_S^{L_S})) \right]$$

However, this approach often fails to capture the rich information encoded within the immediate representations of LLMs. Feature Dynamic Distillation (FDD) (Gong et al., 2025) addresses this limitation by framing the distillation problem as aligning the entire feature dynamics between models, including additional loss term that match both the feature trajectory and its first-order derivative. Our work further extends this approach to a continuous path-based approach to feature-space alignment.

#### 2.2.2. DYNAMIC PERSPECTIVE OF LLMs

A foundational perspective for neural networks is to view their discrete, layer-wise structure as an Euler discretization of a continuous-time system governed by an ordinary differ-

ential equation (ODE) (Chen et al., 2019). In this paradigm, a neural network is not a sequence of layers but a model that parameterizes a continuous vector field. This vector field defines the instantaneous rate of change of the hidden state:

$$\frac{d\mathbf{h}(t)}{dt} = f(h(t), t, \theta)$$

From this dynamic perspective, a student model can be seen as a coarse discretization of the feature dynamics, where a larger model represents a finer representation of the same process (Gong et al., 2025). This suggests that effective knowledge transfer should align the underlying continuous dynamics that govern feature evolution.

### 2.2.3. CONTINUOUS-TIME FLOW-BASED MODELS

Generative modeling with Continuous Normalizing Flows (CNFs) aims to transform a prior distribution  $p_0$  to a complex distribution  $p_1$  via a time-dependent vector field  $u : [0, 1] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ . The flow  $\phi_t(x)$  is defined by an ODE:

$$\frac{d}{dt}\phi_t(x) = u_t(\phi_t(x)), \quad \phi_0(x) = x$$

The probability density  $p_t$  induced by this flow evolves according to the continuity equation

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot (p_t u_t)$$

However, the marginal probability path  $p_t$  is intractable. Flow Matching (Lipman et al., 2023) addresses this challenge by reframing the problem in terms of conditional transport paths between paired samples, rather than explicitly modeling the marginal density evolution. Given a coupling  $\pi(p_0, p_1)$  and a conditional interpolation path  $X_t = \psi_t(x_0, x_1)$  between paired samples  $(x_0, x_1) \sim \pi$ , the associated conditional velocity field is defined as the time derivative of the path,  $u_t(X_t | x_0, x_1) = \frac{d}{dt}\psi_t(x_0, x_1)$ . Flow Matching then trains a neural vector field  $v_\theta$  to regress these conditional velocities along the interpolation, yielding the objective

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1), \\ (x_0, x_1) \sim \pi}} [\|v_\theta(X_t, t) - u_t(X_t | x_0, x_1)\|^2]. \quad (1)$$

Rectified Flow (Liu et al., 2023) arises as a particular instantiation of this framework by choosing the interpolation path to be linear in time,  $X_t = (1-t)x_0 + tx_1$ . Under this choice, the conditional velocity field becomes constant along the trajectory,  $u_t(X_t | x_0, x_1) = x_1 - x_0$ , which reduces the Flow Matching objective to a simple least-squares regression:

$$\mathcal{L}_{\text{RF}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1), \\ (x_0, x_1) \sim \pi}} [\|(x_1 - x_0) - v_\theta(X_t, t)\|^2]. \quad (2)$$

This construction induces straight-line trajectories in Euclidean space and admits exact simulation with a single Euler step, providing a geometrically principled and computationally efficient foundation that we later leverage for feature-space alignment in CONTRA-KD.

## 3. Method

This section details our proposed framework CONTRA-KD. We first show that by learning the flow between teacher and student hidden states, we can distill not only the intermediate representations but also their internal dynamics in Section 3.1. In particular, we will construct the teacher-student flow in Section 3.2. Section 3.3 and 3.4 will elaborate how to apply CONTRA-KD design into distillation process and incorporate with existing methods.

### 3.1. Our Motivation

**Trajectory Alignment Implies Dynamic Alignment.** Under the perspective of continuous-time paradigm, the teacher and student models are dynamic systems over layer index  $t^1$  governed by distinct velocity fields. Let the teacher’s and student’s evolution be governed by velocity fields  $v_T$  and  $v_S$ . Their hidden state trajectories,  $h_T(t)$  and  $h_S(t)^2$ , are solutions of:

$$\frac{dh_T(t)}{dt} = v_T(h_T(t), t); \quad \frac{dh_S(t)}{dt} = v_S(h_S(t), t).$$

To ensure robust knowledge transfer, the student must learn from teacher’s internal dynamics. Mathematically, this is equivalent to minimizing the difference between the trajectories generated by their respective vector fields:

$$\begin{aligned} & \mathbb{E}_{t \sim \mathcal{U}(0,1)} [\|h_T(t) - h_S(t)\|_{\mathcal{L}_2}] \\ &= \mathbb{E}_{t \sim \mathcal{U}(0,1)} \left[ \left[ \int_{\Omega} [h_T(t; x, y) - h_S(t; x, y)]^2 dx dy \right]^{1/2} \right], \end{aligned}$$

where  $\Omega$  is the domain of  $(x, y)$ .

Let us denote  $\delta_t = v_S(h_T(t), t) - v_T(h_T(t), t)$ . In the following theorem, we prove that the trajectory distillation can lead to the dynamic or velocity distillation.

**Theorem 3.1.** *Assume that the teacher and student velocities satisfy  $M$ -Lipchitz. We have the following inequality:*

$$\begin{aligned} \|\delta_t\|_{\mathcal{L}_2} &\leq M \|h_T(t) - h_S(t)\|_{\mathcal{L}_2} \\ &\quad + \left\| \frac{d}{dt} [h_S(t) - h_T(t)] \right\|_{\mathcal{L}_2}, \end{aligned} \quad (3)$$

<sup>1</sup>Here,  $t$  implies the continuous time variable over which the hidden representations transform; it is equivalent to the discrete layer indices of Transformer models.

<sup>2</sup> $h_T(t)$  and  $h_S(t)$  are indeed  $h_T(x, y, t)$  and  $h_S(x, y, t)$  where  $(x, y)$  represents a pair of input and output sampled from the ground-truth data distribution. We ignore them for brevity.

where  $\|f\|_{\mathcal{L}_2} := \left( \int f(a)^2 da \right)^{1/2}$ .

The detailed theoretical proof is provided in Appendix A. Equation (3) shows a direction connection between trajectory alignment and velocity alignment, which bounds the dynamic difference  $\delta_t$  in terms of the trajectories and its rate of change. When the student is trained to distill the teacher’s hidden states, the state difference  $\|h_T(t) - h_S(t)\|_{\mathcal{L}_2}$  and its temporal derivative  $\|\frac{d}{dt}[h_S(t) - h_T(t)]\|_{\mathcal{L}_2}$  are driven to 0. Consequently, the dynamic or velocity difference  $\delta_t$  also converges to 0. This implies that matching the trajectories implicitly enforces the student’s dynamics to follow the teacher’s dynamics. Therefore, trajectory distillation leads to dynamic distillation, ensuring the student not only reproduces the teacher’s states but also learns its underlying dynamics.

**Curriculum Learning through Continuous Flow.** While theorem 3.1 shows that aligning student and teacher trajectories implicitly enforces dynamic alignment, directly matching trajectories can be difficult, especially at early training stages when the student’s representations are far from teacher’s. This motivates a curriculum-based strategy, where the student is not forced to immediately match the teacher’s states but instead progress toward them through a sequence of intermediate targets (Figure 1). To realize this curriculum, we define a straight flow connecting corresponding representations of student and teacher, where intermediate states of this flow serve as progressively easier alignment targets. As training progresses, the step size is gradually increased, enables the student to traverse longer segments and progressively approach teacher’s trajectory. Further analysis can be found in Section 5.1.

### 3.2. Student-Teacher Continuous Flow

Formally, let  $h_S^j \in \mathbb{R}^{N \times d_S}$  and  $h_T^j \in \mathbb{R}^{N \times d_T}$  be the feature representations coming out from a selected intermediate layer  $j$  of a student and teacher model, respectively, for a given input with sequence length  $N$ . Typically, they reside in spaces of different dimensions. With the assumption that the teacher’s hidden space is better trained and can express more information, we first project the student’s hidden states to the teacher’s space and use these projected hidden states for later steps. This projection can be parameterized as simple as a weight matrix  $W^{S \rightarrow T} \in \mathbb{R}^{d_S \times d_T}$ :

$$f_{\text{proj}}(h_S^j) := h_S^j W^{S \rightarrow T} \in \mathbb{R}^{N \times d_T}$$

A second challenge is the mismatch in the number of layers ( $L_S \neq L_T$ ). To tackle this, we use a uniform layer scheduling schema by selecting a subset of  $L_D$  layers from both the student and the teacher and form pairs of aligned layers. We denote  $I_S = \{l_1^S, l_2^S, \dots, l_{L_D}^S\}$  and  $I_T = \{l_1^T, l_2^T, \dots, l_{L_D}^T\}$

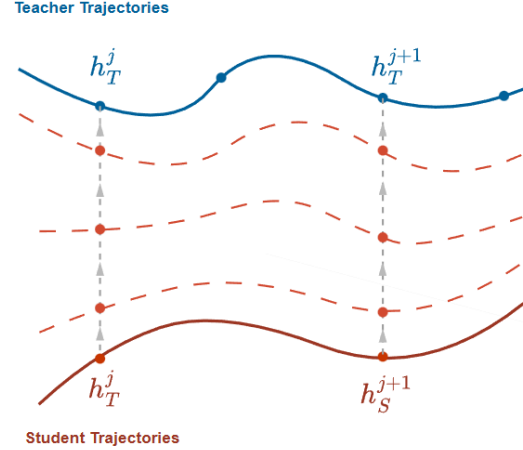


Figure 1. Curriculum learning through Continuous Flow. The blue line denotes teacher trajectories, the red line denotes student trajectories, and the dotted gray arrow indicates our learned student-teacher flow. In early step of training, the student follows a short segment of this flow with an intermediate target that is easier to optimize, and gradually increases the step sizes as training progresses.

as the ordered sets of selected layer indices for the student and the teacher, respectively.

The transformation is modeled as the trajectory of a time-dependent ODE, where for the  $i$ -th token and the  $j$ -th layer pair in our schedule, a continuous state  $Z_t^{(i,j)}$  evolves from  $t = 0$  to  $t = 1$  according to:

$$\frac{dZ_t^{(i,j)}}{dt} = v_\theta(Z_t^{(i,j)}, t, j).$$

The trajectory starts at the student’s distribution  $Z_0^{(i,j)}$  and is guided to the teacher’s distribution  $Z_1^{(i,j)}$ . We denote  $Z_0^{(i,j)} := h_S^j[i]$  as the student’s hidden representation of the  $i$ -th token in the token sequence at the  $j$ -th aligned layer (i.e., the  $l_j^S$ -th layer in the student model) and, similarly,  $Z_1^{(i,j)} := h_T^j[i]$  as the teacher’s hidden representation of the same token at the teacher’s  $l_j^T$ -th layer. The neural network  $v_\theta$ , conditioned on distillation step index  $j$ , parameterizes the dynamics of this transformation, learning the direction and magnitude of feature correction at any intermediate point ( $Z_t$ ) along the path.

In Euclidean geometry of feature spaces, the straight path is the shortest possible route between two endpoints and can be simulated without time discretization. Therefore, we constrain our learned ODE to follow this trajectory. To achieve this, we apply the learning objective of Rectified Flow. For any pair of corresponding states  $(h_S^j[i], h_T^j[i])$ ,

**Algorithm 1** Training the Velocity Field  $v_\theta$ 


---

```

1: Input: teacher  $p$ , student  $q$ , projector  $f_{\text{proj}}$ , velocity
   field  $v_\theta$  with parameter  $\theta$ , data  $\mathcal{D}$ , layer maps  $I^S, I^T$ 
2: Output: Trained velocity field  $v_\theta$ 
3: for each batch  $\{x_i\}$  in  $\mathcal{D}$  do
4:   Initialize  $\mathcal{L}(\theta) \leftarrow 0$ 
5:   for  $j = 1, 2, \dots, L_D$  do
6:      $h_S \leftarrow f_{\text{proj}}(q(\{x_i\}, \text{layer} = j))$ 
7:      $h_T \leftarrow p(\{x_i\}, \text{layer} = I_j^T)$ 
8:      $t \sim U(0, 1)$ 
9:      $H_t \leftarrow (1 - t) \cdot h_S + t \cdot h_T$ 
10:     $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\theta) + \|(h_T - h_S) - v_\theta(H_t, t, j)\|^2$ 
11:  end for
12:  Update  $\theta$  using gradients from  $\mathcal{L}(\theta)$ 
13: end for
14: return  $v_\theta$ 

```

---

we define a linear interpolant  $H_t^{(i,j)}$ :

$$H_t^{(i,j)} := (1 - t) h_S^j[i] + t h_T^j[i].$$

The target velocity vector required to traverse this path is a constant vector, that is, the vector from the student's state to the teacher's state:  $\frac{dH_t^{(i,j)}}{dt} = h_T^j[i] - h_S^j[i]$ . Our goal is to train a velocity field network  $v_\theta$  that perfectly replicates this target vector field. The learning objective for  $v_\theta$  is as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{i,j,t} \left[ \left\| (h_T^j[i] - h_S^j[i]) - v_\theta(H_t^{(i,j)}, t, j) \right\|^2 \right].$$

The expectation is taken over sequence of tokens  $i$ , distillation steps  $j \in \{1, \dots, L_D\}$ , and time steps  $t \sim U(0, 1)$ . The overall training procedure of the velocity field is detailed in Algorithm 1

### 3.3. Curriculum-based Trajectory Distillation

**Curriculum-based Approach.** Directly distilling the teacher trajectory by alignment between  $h_S(t)$  and  $h_T(t)$  can be challenging at early stages of training, where the student representation is still far from the teacher's and can lead to unstable update signals. To address this issue, we introduce a **curriculum-based trajectory distillation** strategy that effectively utilizes supervision from the learned flow.

Specifically, we first define an intermediate target  $\tilde{h}_S$  along the velocity field  $v_\theta$  such that  $\tilde{h}_S = h_{S\#} \int_0^k v_\theta(t) dt$  where  $k \in (0, 1]$ . Intuitively,  $\tilde{h}_S$  lies in partway along the learned flow induced by  $v_\theta$  at distance defined by time step  $k$  rather than its endpoint. The target can be effectively approximated by:

$$\tilde{h}_S = h_S + \int_0^k v_\theta(t) dt$$

**Algorithm 2** CONTRA-KD Distillation

---

```

1: Input: student  $q_{\phi_0}$  with parameters  $\phi_0$ , teacher  $p$ , pre-
   trained velocity field  $v_\theta$  and projector  $f_{\text{proj}}$ , data  $\mathcal{D}$ , layer
   maps  $I^S, I^T$ 
2: Output: Student model  $q_{\phi_E}$  with trained parameters
    $\phi_E$ 
3: for epoch  $e = 1, 2, \dots, E$  do
4:    $k \leftarrow \frac{e}{E}$ 
5:   for each batch  $\{x_i\}$  in  $\mathcal{D}$  do
6:     // — Phase 1: Update Student Model —
7:     Initialize  $\mathcal{L}_{\text{CONTRA}} \leftarrow 0$ 
8:     for  $j = 1, 2, \dots, L_D$  do
9:        $h_S \leftarrow f_{\text{proj}}(q_{\phi_{e-1}}(\{x_i\}, \text{layer} = j))$ 
10:       $\tilde{h}_S \leftarrow h_S + k \cdot v_\theta(h_S, t = 0, j)$ 
11:       $\mathcal{L}_{\text{CONTRA}} \leftarrow \mathcal{L}_{\text{CONTRA}} + D_{\text{cos}}(h_S, \tilde{h}_S)$ 
12:    end for
13:    Compute  $\mathcal{L}_{KD}$  using  $p(x_i)$  and  $q_{\phi_{e-1}}(x_i)$ 
14:     $\mathcal{L}_{\text{Total}} \leftarrow \mathcal{L}_{CE} + \mathcal{L}_{KD} + \mathcal{L}_{\text{CONTRA}}$ 
15:    Update  $\phi_e \leftarrow \phi_{e-1} - \eta \nabla_{\phi} \mathcal{L}_{\text{Total}}$ 
16:  end for
17:  // — Phase 2: Adapt Velocity Field & Projector —
18:  Update velocity field  $v_\theta$  and projector  $f_{\text{proj}}$  based on
   Algorithm 1 using teacher  $p$ , student  $q_{\phi_e}$ , data  $\mathcal{D}$ ,
   layer maps  $I^S, I^T$ 
19: end for
20: return  $q_{\phi_E}$ 

```

---

For smaller value of  $k$ , the target is easier to follow, forming a curriculum that progressively increases in difficulty as training proceeds.

**Distillation Process.** Let  $h_S^j[i]$  denote the projected hidden representation of sample  $i$  at the  $j$ -th aligned layer. The corresponding curriculum-based target is defined as

$$\tilde{h}_S^j[i] = h_S^j[i] + k \cdot v_\theta(h_S^j[i], t = 0, j).$$

During training,  $k$  is gradually increased based on training iterations:  $k = \frac{t}{T}$ , where  $t$  is the iteration index and  $T$  is the total number of training iterations. The CONTRA loss then minimizes the discrepancy between the student's layer output and this dynamically generated target:

$$\mathcal{L}_{\text{CONTRA}} = \frac{1}{N L_D} \sum_{i=1}^N \sum_{j=1}^{L_D} D_{\text{cos}}(h_S^j[i], \tilde{h}_S^j[i]),$$

where  $D_{\text{cos}}(a, b) = 1 - \frac{a^T \cdot b}{\|a\| \|b\|}$  is the cosine distance between 2 vectors. We adopt the cosine distance as it emphasizes directional alignment of student updates with the rectified flow. To avoid distribution shift when updating the student model, we continue to train the velocity field and the projector after each distillation epoch. The distillation process can be found in details in Algorithm 2.

Table 1. Win / Tie / Loss rates (%) across student-teacher pairs on different benchmark datasets, evaluated by GPT-4o-mini. **Win** indicates preference for the model trained with CONTRA over the baseline (Ours Win), **Loss** indicates preference for the baseline (Baseline Win), and **Tie** denotes equal quality.

Method	Evaluation Set								
	Dolly			Self-Instruct			Vicuna		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
<i>LLaMA2 13B → 7B</i>									
FDD (+CONTRA vs. baseline)	<b>44.00</b>	19.60	36.40	38.43	23.14	38.43	<b>48.75</b>	10.00	41.25
DistiLLM (+CONTRA vs. baseline)	<b>41.20</b>	22.60	36.20	<b>43.80</b>	14.40	41.80	<b>46.25</b>	15.00	38.75
DistiLLM-2 (+CONTRA vs. baseline)	<b>40.40</b>	26.60	33.00	<b>43.80</b>	15.29	40.91	<b>45.00</b>	21.25	33.75
<i>OpenLLaMA2 7B → 3B</i>									
FDD (+CONTRA vs. baseline)	<b>41.00</b>	18.40	40.60	<b>48.35</b>	9.09	42.56	<b>55.00</b>	5.00	40.00
DistiLLM (+CONTRA vs. baseline)	<b>44.00</b>	14.80	41.20	<b>45.00</b>	12.50	42.50	<b>48.75</b>	10.00	41.25
DistiLLM-2 (+CONTRA vs. baseline)	<b>43.00</b>	34.60	22.40	<b>46.69</b>	13.64	39.67	<b>53.75</b>	7.50	38.75
<i>GPT2 1.5B → 0.1B</i>									
FDD (+CONTRA vs. baseline)	<b>40.00</b>	25.00	35.00	<b>47.52</b>	16.94	35.54	<b>42.50</b>	22.50	35.00
DistiLLM (+CONTRA vs. baseline)	<b>41.40</b>	21.20	37.40	<b>33.30</b>	10.20	32.60	<b>40.00</b>	22.50	37.50
DistiLLM-2 (+CONTRA vs. baseline)	<b>42.80</b>	22.60	34.60	<b>40.91</b>	28.92	30.17	<b>37.50</b>	31.25	31.25

### 3.4. Final Objective Function

A key advantage of CONTRA-KD is its orthogonality to existing distillation objectives. It acts as a compliment on internal feature space, while other methods utilizing KD loss terms focus on output logits. Therefore, we integrate  $\mathcal{L}_{\text{CONTRA}}$  as a complementary term to standard KD baselines. The final loss function is a weighted sum of these components:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{KD} + \mathcal{L}_{CONTRA}$$

## 4. Experiments

### 4.1. Experimental Setup

**Datasets.** We evaluate the effectiveness of our proposed framework through extensive experiments on instruction-following benchmarks. We include sampled test set from *databricks-dolly-15k* dataset, Self-Instruct (Wang et al., 2023), Vicuna (Chiang et al., 2023), Super-Natural Instructions (S-NI) (Wang et al., 2022), Unnatural Instructions (UnNI) (Honovich et al., 2022).

**Baselines.** To demonstrate the versatility and complementary nature of our framework, we evaluate CONTRA-KD as an additive objective that integrates with existing distillation methods. We compare the performance of state-of-the-art baselines, specifically DistiLLM (Ko et al., 2024), DistiLLM-2 (Ko et al., 2025), and FDD (Gong et al., 2025), both in their standard standalone configurations and when augmented with our CONTRA-KD auxiliary loss. This setup allows us to isolate the specific gains attributed to continuous trajectory alignment.

**Base Models.** Our method is evaluated on three pairs of

teacher-student models: GPT-2 (Radford et al., 2019) (1.5B teacher and 0.1B student), LLaMA2 (Touvron et al., 2023) (13B teacher and 7B student), and OpenLLaMA2 (Geng & Liu, 2023) (7B teacher and 3B student). Details on training setups can be found in Appendix B.

**Evaluation.** We employ LLM-as-a-judge to evaluate the quality of responses sampled from models trained using baseline methods and those trained with CONTRA-KD included to observe the improvements in performance that CONTRA-KD can bring to state-of-the-art KD methods. Specifically, we use GPT-4o-mini as the judge, following prior work on LLM-based evaluation (Zheng et al., 2023), for pairwise comparison of responses, comparing the baseline’s response and CONTRA-KD’s response for a given input prompt. We record win/tie/loss rates on the Dolly test set, Self-Instruct, and Vicuna. Additionally, we record ROUGE-L scores of models’ generated responses compared to ground-truth ones over the 5 instruction-following datasets.

### 4.2. Results

We present the evaluation results across the instruction-following datasets in Table 1 and 2. CONTRA-KD consistently achieves the highest average ROUGE-L scores in all settings, improving upon both the adaptive divergence method (DistiLLM) and the discrete feature matching method (FDD). On the **OpenLlama2-7B → OpenLlama2-3B** pair, adding CONTRA-KD to DistiLLM results in an average gain of **+0.81%**, securing the best performance across every single dataset. Similarly, in the high-compression setting of **GPT-2 1.5B → 0.1B**, our method boosts the DistiLLM baseline by **+1.22%** on average. Most impressively,

on the UnNI dataset for the Llama2-13B pair, CONTRA-KD guides the student to a score of **37.70**, surpassing the teacher model itself (34.25).

Consistent with the automatic metrics, GPT-4o-mini evaluations show that our method achieves higher win rates against all baselines across all settings, confirming that the ROUGE-L improvements correspond to better instruction-following quality. Overall, these results demonstrate the robustness and effectiveness of our framework across diverse datasets, model architectures, and compression ratios.

Table 2. The effects of CONTRA-KD on the SOTA methods. Rouge-L scores (%) averaged over 5 random seeds.

Method	Dolly	Self-Inst	Vicuna	S-NI	UnNI	Avg.
<i>LLaMA2-13B → LLaMA2-7B</i>						
Teacher	30.19	23.92	19.56	34.53	34.25	28.49
SFT	28.38	20.32	17.72	34.37	33.25	26.81
DistiLLM	31.10	23.22	21.38	<b>36.30</b>	37.03	29.81
+CONTRA	<b>31.45</b>	<b>23.69</b>	<b>21.48</b>	36.19	<b>37.70</b>	<b>30.10</b>
FDD	30.59	22.12	20.87	35.91	36.48	29.19
+CONTRA	<b>31.55</b>	<b>22.80</b>	<b>20.92</b>	<b>36.81</b>	<b>37.08</b>	<b>29.83</b>
DistiLLM2	31.76	<b>24.49</b>	21.96	35.98	36.57	30.15
+CONTRA	<b>32.19</b>	23.97	<b>22.27</b>	<b>36.52</b>	<b>38.59</b>	<b>30.71</b>
<i>OpenLLaMA2-7B → OpenLLaMA2-3B</i>						
Teacher	27.25	19.19	17.69	31.51	31.05	25.34
SFT	24.95	17.38	14.58	28.65	27.69	22.65
DistiLLM	28.97	19.39	19.60	34.46	34.26	27.34
+CONTRA	<b>29.04</b>	<b>20.20</b>	<b>19.66</b>	<b>35.65</b>	<b>36.20</b>	<b>28.15</b>
FDD	28.07	<b>19.47</b>	19.33	35.47	34.83	27.43
+CONTRA	<b>28.31</b>	19.12	<b>20.00</b>	<b>36.56</b>	<b>36.05</b>	<b>28.01</b>
DistiLLM2	28.48	19.78	19.85	35.66	36.52	28.06
+CONTRA	<b>29.44</b>	<b>20.38</b>	<b>20.59</b>	<b>35.74</b>	<b>36.80</b>	<b>28.59</b>
<i>GPT2 1.5B → GPT2 0.1B</i>						
Teacher	27.22	13.38	16.33	26.90	30.38	22.84
SFT	23.33	10.56	15.12	17.08	20.07	17.23
DistiLLM	25.17	11.93	16.63	23.29	25.41	20.49
+CONTRA	<b>26.22</b>	<b>12.99</b>	<b>17.41</b>	<b>25.07</b>	<b>26.85</b>	<b>21.71</b>
FDD	24.27	11.23	14.69	19.65	22.50	18.47
+CONTRA	<b>25.12</b>	<b>11.46</b>	<b>16.05</b>	<b>20.21</b>	<b>22.79</b>	<b>19.13</b>
DistiLLM2	22.22	11.13	17.19	21.58	24.39	19.30
+CONTRA	<b>22.56</b>	<b>11.74</b>	<b>18.10</b>	<b>22.09</b>	<b>24.92</b>	<b>19.88</b>

## 5. Analysis

We provide ablation studies on CONTRA-KD, which are experimented with GPT-2 1.5B → 0.1B teacher-student pair settings and assess the results with ROUGE-L metric. Additional results can be found at Appendix C.

### 5.1. Impact of Curriculum-based Distillation

We analyze the effect of curriculum-based step sizes in CONTRA-KD by comparing our progressive step schedule against directly applying full update throughout training (i.e.

applying fixed step size  $k = 1$ ). In the progressive setting, the step size  $k$  is gradually increased from small values to 1 over the course of training, allowing the student to follow increasingly longer segments of the learned flow.

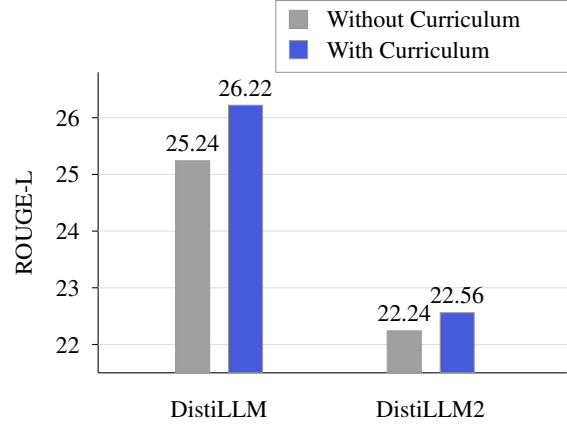


Figure 2. Comparison between curriculum-based and full-step distillation for hidden-state targets.

As shown in Figure 2, curriculum-based distillation consistently improves performance across both DistiLLM and DistiLLM2 baselines. This highlights well-conditioned intermediate targets at early stages of training and gradually increasing the difficulty of the alignment task enable more effective trajectory alignment, resulting in higher ROUGE-L scores.

### 5.2. Computational Cost

We analyze the computational overhead introduced by adding CONTRA by comparing the average wall-clock training time per epoch across different distillation settings in Table 3. Overall, incorporating CONTRA leads to a moderate and consistent increase in training time, with the relative overhead remaining below  $1.5\times$  in all configurations. Overall, these results show that while CONTRA introduces a consistent increase in training cost, the overhead remains controlled across scales and methods, supporting its practicality for large-scale distillation settings.

### 5.3. Impact of Number of Distilled Layers

We study the effect of the number of distilled layers  $L_D$  in CONTRA-KD, which controls how many intermediate student-teacher layer pairs are used to learn and apply the trajectory alignment. A larger  $L_D$  provides denser supervision along the feature trajectory, while a smaller  $L_D$  reduces computational overhead but offers coarser guidance.

Figure 3 illustrates how the number of distilled intermediate layers influences student performance. Performance peaks when a moderate number of four layers is distilled, after which further increasing the number of layers results in a

Table 3. Training cost analysis. We report the average wall-clock time per epoch in minutes.

Method	Standard	+CONTRA	Increase
<i>GPT2 1.5B <math>\rightarrow</math> 0.1B</i>			
DistiLLM	13.50	14.15	1.05 $\times$
FDD	9.25	12.45	1.35 $\times$
DistiLLM2	12.20	13.05	1.07 $\times$
<i>OpenLLaMA2 7B <math>\rightarrow</math> 3B</i>			
DistiLLM	28.50	32.90	1.15 $\times$
FDD	23.80	26.90	1.13 $\times$
DistiLLM2	65.90	78.60	1.19 $\times$
<i>LLaMA2 13B <math>\rightarrow</math> 7B</i>			
DistiLLM	32.50	43.10	1.33 $\times$
FDD	27.20	39.80	1.46 $\times$
DistiLLM2	71.60	89.30	1.25 $\times$

slight degradation. Following the conclusion of (Gong et al., 2025), we attribute this decline to over-regularization, where excessive constraints limit the student’s ability to adapt its own dynamics.

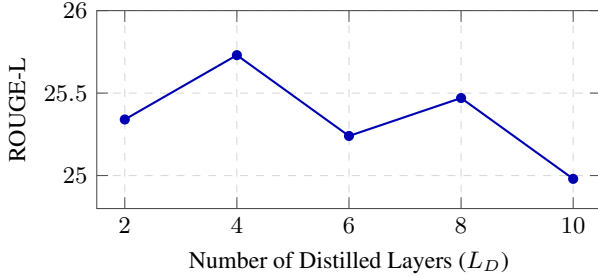


Figure 3. Impact of the number of distilled layers  $L_D$  on distillation performance.

#### 5.4. Non-linear Interpolant Path Analysis

A core efficiency mechanism of CONTRA-KD is the generation of the dynamic distillation target via an explicit Euler update along a learned flow (see Section 3.3). This approximation is geometrically faithful when the underlying flow trajectory is linear, but may incur truncation error when applied to curved paths. To validate this design choice, we compare our linear Rectified Flow against two non-linear path formulations commonly used in diffusion models: Variance Preserving (VP) ODE (Song et al., 2021) and Variance Exploding (VE) ODE (Song et al., 2021).

The flow trajectories are defined by a generalized interpolant  $H_t$  between the student state  $h_S^j[i]$  and teacher state  $h_T^j[i]$ :

$$H_t = \sigma_t h_S^j[i] + \alpha_t h_T^j[i] \quad (4)$$

where  $\sigma_t$  and  $\alpha_t$  control the mixing schedule. We compare three schedules:

Table 4. Ablation study on the impact of interpolation path geometry (GPT-2 1.5B  $\rightarrow$  0.1B). The linear path (Rectified Flow) outperforms non-linear schedules.

Interpolation Path	ROUGE-L
Rectified Flow (Ours)	<b>25.24</b>
VP-ODE	24.71
VE-ODE	25.23

- **Rectified Flow (Ours):** The linear path with  $\alpha_t = t$  and  $\sigma_t = 1 - t$ .
- **VP ODE:** A cosine-based schedule defined by  $\alpha_t = \exp(-\frac{1}{4}a(1-t)^2 - \frac{1}{2}b(1-t))$  and  $\sigma_t = \sqrt{1 - \alpha_t^2}$ , with  $a = 19.9$  and  $b = 0.1$ .
- **VE ODE:** A geometric variance schedule defined by  $\alpha_t = a(\frac{b}{a})^t$  and  $\sigma_t = 1$ , with  $a = 0.02$  and  $b = 100$ .

The results in Table 4 show that the VP-ODE path causes a noticeable performance drop ( $-0.53$  ROUGE-L). This indicates that curved interpolation paths are ill-suited for Euler-based target generation, as local curvature introduces truncation error even when using partial steps along the flow. While VE-ODE performs comparably to the linear baseline, it requires careful hyperparameter tuning ( $a, b$ ) and offers no performance gain. In contrast, the linear Rectified Flow consistently provides a stable and accurate approximation under Euler updates, making it the most effective choice for curriculum-based trajectory distillation.

## 6. Conclusion

In this work, we presented **CONTRA-KD**, a novel knowledge distillation framework that differs from layer-wise matching by explicitly modeling distillation as continuous-time alignment problem. By interpreting both teacher and student models as dynamical systems, we proved that minimizing the discrepancy between student and teacher trajectories implicitly bounds the difference between their corresponding velocity fields. Building on this insight, we use a rectified flow formulation that learns a transformation between student and teacher hidden states and employ a curriculum-based strategy to progressively guide the student along this flow. Extensive experiments across multiple model pairs on instruction-following benchmarks demonstrate the effectiveness and compatability of CONTRA-KD while maintaining reasonable computation overhead. Overall, our findings highlight the value of continuous-time and dynamical-system perspectives and suggest promising directions for leveraging trajectory alignment and flow-based methods in knowledge distillation.

## Impact Statement

This paper presents a methodological contribution to the field of machine learning, focusing on improving the efficiency and stability of knowledge distillation for large language models. By framing distillation through continuous-time dynamics and trajectory alignment, our work aims to facilitate the transfer of capabilities from large models to smaller, more computationally efficient ones. Such techniques can help reduce the computational and environmental costs associated with deploying large-scale models, potentially enabling broader access to advanced language technologies.

While we have illustrated the capability of our work in language domain, we believe it can be applied in broader fields. We encourage future potential research to discuss the applications of this work across domains, such as multi-modal LLMs.

## References

- Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Ramos, S., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes, 2024. URL <https://arxiv.org/abs/2306.13649>.
- Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016. URL <https://arxiv.org/abs/1607.06450>.
- Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations, 2019. URL <https://arxiv.org/abs/1806.07366>.
- Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality. <https://lmsys.org/blog/2023-03-30-vicuna/>, 2023.
- Geng, X. and Liu, H. OpenLLaMA: An open reproduction of LLaMA. [https://github.com/openlm-research/open\\_llama](https://github.com/openlm-research/open_llama), 2023. Software release.
- Gokaslan, A. and Cohen, V. Openwebtext corpus. <http://Skylion007.github.io/OpenWebTextCorpus>, 2019.
- Gong, G., Wang, J., Xu, J., Xiang, D., Zhang, Z., Shen, L., Zhang, Y., JunhuaShu, J., ZhaolongXing, Z., Chen, Z., Liu, P., and Zhang, K. Beyond logits: Aligning feature dynamics for effective knowledge distillation. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23067–23077, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1125. URL <https://aclanthology.org/2025.acl-long.1125/>.
- Gu, Y., Dong, L., Wei, F., and Huang, M. Minillm: Knowledge distillation of large language models, 2024. URL <https://arxiv.org/abs/2306.08543>.
- Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus), 2023. URL <https://arxiv.org/abs/1606.08415>.
- Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. <https://arxiv.org/abs/1503.02531>, 2015. arXiv preprint arXiv:1503.02531.
- Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor, 2022. URL <https://arxiv.org/abs/2212.09689>.
- Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. Tinybert: Distilling bert for natural language understanding, 2020. URL <https://arxiv.org/abs/1909.10351>.
- Kim, Y. and Rush, A. M. Sequence-level knowledge distillation, 2016. URL <https://arxiv.org/abs/1606.07947>.
- Ko, J., Kim, S., Chen, T., and Yun, S.-Y. Distillm: Towards streamlined distillation for large language models, 2024. URL <https://arxiv.org/abs/2402.03898>.
- Ko, J., Chen, T., Kim, S., Ding, T., Liang, L., Zharkov, I., and Yun, S.-Y. Distillm-2: A contrastive approach boosts the distillation of llms, 2025. URL <https://arxiv.org/abs/2503.07067>.
- Lin, A., Wohlwend, J., Chen, H., and Lei, T. Autoregressive knowledge distillation through imitation learning, 2020. URL <https://arxiv.org/abs/2009.07253>.
- Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://aclanthology.org/W04-1013/>.

- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling, 2023. URL <https://arxiv.org/abs/2210.02747>.
- Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2023. URL <https://arxiv.org/abs/2209.03003>.
- Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T.-Y. Understanding and improving transformer from a multi-particle dynamic system point of view, 2019. URL <https://arxiv.org/abs/1906.02762>.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. [https://cdn.openai.com/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf), 2019. OpenAI technical report. Accessed: 2024-11-15.
- Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL <https://arxiv.org/abs/1910.01108>.
- Skean, O., Arefin, M. R., Zhao, D., Patel, N., Naghiyev, J., LeCun, Y., and Shwartz-Ziv, R. Layer by layer: Uncovering hidden representations in language models, 2025. URL <https://arxiv.org/abs/2502.02013>.
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations, 2021. URL <https://arxiv.org/abs/2011.13456>.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models. <https://arxiv.org/abs/2302.13971>, 2023. arXiv preprint arXiv:2302.13971.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
- Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., Pathak, E., Karamanolakis, G., Lai, H. G., Purohit, I., Mondal, I., Anderson, J., Kuznia, K., Doshi, K., Patel, M., Pal, K. K., Moradshahi, M., Parmar, M., Purohit, M., Varshney, N., Kaza, P. R., Verma, P., Puri, R. S., Karia, R., Sampat, S. K., Doshi, S., Mishra, S., Reddy, S., Patro, S., Dixit, T., Shen, X., Baral, C., Choi, Y., Smith, N. A., Hajishirzi, H., and Khashabi, D. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks, 2022. URL <https://arxiv.org/abs/2204.07705>.
- Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions, 2023. URL <https://arxiv.org/abs/2212.10560>.
- Wu, T., Tao, C., Wang, J., Yang, R., Zhao, Z., and Wong, N. Rethinking kullback-leibler divergence in knowledge distillation for large language models, 2024. URL <https://arxiv.org/abs/2404.02657>.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL <https://arxiv.org/abs/2306.05685>.

## A. Theory Development

In this section, we will give mathematical proof showing that by minimizing the continuous trajectory error, we implicitly constrain the student’s governing vector field  $v_S$  to align with the effective dynamics  $v_T$  of the teacher.

*Proof.* We start with

$$\begin{aligned}\delta_t &= v_S(h_T(t), t) - v_T(h_T(t), t) \\ &= [v_S(h_T(t), t) - v_S(h_S(t), t)] \\ &\quad + [v_S(h_S(t), t) - v_T(h_T(t), t)].\end{aligned}$$

This leads to

$$\begin{aligned}\|\delta_t\|_{\mathcal{L}_2} &\leq \|v_S(h_T(t), t) - v_S(h_S(t), t)\|_{\mathcal{L}_2} \\ &\quad + \|v_S(h_S(t), t) - v_T(h_T(t), t)\|_{\mathcal{L}_2} \\ &= \|v_S(h_T(t), t) - v_S(h_S(t), t)\|_{\mathcal{L}_2} \\ &\quad + \left\| \frac{d}{dt} [h_S(t) - h_T(t)] \right\|_{\mathcal{L}_2}.\end{aligned}$$

Using the  $M$ -Lipchitz, we have

$$\|v_S(h_T(t), t) - v_S(h_S(t), t)\|_{\mathcal{L}_2} \leq M \|h_T(t) - h_S(t)\|_{\mathcal{L}_2}$$

Finally, we reach the conclusion

$$\begin{aligned}\|\delta_t\|_{\mathcal{L}_2} &\leq M \|h_T(t) - h_S(t)\|_{\mathcal{L}_2} \\ &\quad + \left\| \frac{d}{dt} [h_S(t) - h_T(t)] \right\|_{\mathcal{L}_2}.\end{aligned}\tag{5}$$

□

## B. Detailed Experimental Setup

For training as a neural network, we design the velocity field  $v_\theta$  using components based on the design of Transformers (Vaswani et al., 2023). We use 2 embedding layers to encode the time step and layer step information, along with an input projection to reduce the model’s hidden dimension size. Then, the embedded input goes through a sequence of feed-forward blocks, each constructed similar to the feed-forward network in a Transformer block. The representations first go through a layer normalization module (Ba et al., 2016) to reduce variance, allowing the training process to be more stable; then, they go through 2 linear layers with a GELU (Hendrycks & Gimpel, 2023) activation and the in-between dimension size of 4 times the model’s hidden dimension size. Finally, the representation vectors go through a dropout module to ensure robustness of the network, before passing through an output projection back to the teacher’s space.

**Implementation Details.** Our experiments are based on previous implementation setups by (Gu et al., 2024; Gong et al., 2025). We use a preprocessed version of *databricks-dolly-15k*, consisting of around 11,500 samples for training, 1000 samples for validation, and 500 samples for testing. Along with that, we include a language modeling loss term on a small subset sampled from the OpenWebText (Gokaslan & Cohen, 2019) corpus, as it has been shown by previous works to improve instruction-tuning performance.

The student models are first fine-tuned for 3 epochs on the training set and used along with the corresponding teacher models for the initial training process of the velocity field. Then, the trained velocity fields act as initial checkpoints to be used for knowledge distillation, where they are updated along with the student models to avoid distributional shift. Student models are validated during the distillation stage by evaluating their generated responses against the reference responses on the validation set using ROUGE-L (Lin, 2004), and the checkpoints with the best results are selected for later evaluation on benchmark datasets.

**Training configurations.**

- **Velocity field config.** We train the velocity field for 3 epochs in all of our experiments. To stay consistent between pairs of teacher-student, we fix the number of feed-forward blocks of the velocity field model to 4 and distill from 6 teacher layers. We set the model’s hidden dimension size to 1024, 2400, and 3200 for GPT-2, OpenLLaMA2, and LLaMA2 models, respectively. This keeps the velocity fields compact while still allowing for distillation between larger models.
- Across all stages of our experiments, we use the same learning rate of  $5e-4$  and effective batch size of 16. For GPT-2 models, we employ full fine-tuning for 20 epochs, except for the teacher GPT2 which is trained for only 10 epochs; for LLaMA and OpenLLaMA models, we fine-tune using LoRA (Hu et al., 2022) for 10 epochs with the rank and alpha both set at 16 and the dropout rate at 0.1.
- **Evaluation configurations.** LLM-as-a-judge: sample 1 response per input prompt; for each baseline and input prompt, ask GPT-4o-mini to compare response from baseline without CONTRA loss and baseline with CONTRA loss using prompt format in 4; results are recorded for each benchmark dataset. ROUGE-L: sample responses using 5 random seeds {10, 20, 30, 40, 50} and take average over the 5 seeds for each benchmark dataset. General: We set the maximum number of tokens generated to 512, the decoding temperature to 1.0 and top\_p to 1.0.

Please act as an impartial judge and compare the quality of response A and response B provided by two AI assistants to the user question displayed below.  
 Focus mainly on how natural, fluent, and human-like the language sounds. Prefer responses that are short, direct, and easy to understand rather than those that are long, formal, or overly detailed.  
 Your evaluation should prioritize clarity, simplicity, and readability over technical accuracy, completeness.

If A is significantly better overall, answer “A”.  
 If B is significantly better overall, answer “B”.  
 If both are similar in quality (both bad or both good or not significantly different), answer “Tied”.

[Question]  
 {}

[Response A]  
 {}

[Response B]  
 {}

Figure 4. Prompt used for pairwise response quality evaluation.

## C. Additional Ablation Studies

### C.1. Impact of Velocity Field Architecture

We analyze the impact of the velocity field architecture on distillation performance, focusing on two key design factors: network depth and hidden dimension. All experiments are conducted on the Dolly test set using the GPT-2 1.5B  $\rightarrow$  0.1B distillation setting.

**Impact of velocity field depth.** We first examine the effect of network depth by varying the number of feed-forward blocks in the velocity field network  $v_\theta$  from 2 to 6, while keeping the hidden dimension fixed at 1024. As shown in Figure 5 (left), increasing the depth of the velocity field does not improve performance; instead, it leads to a gradual degradation. The highest ROUGE-L score of **25.58** is achieved with the shallowest configuration (2 layers), whereas the 6-layer model drops to 24.80. This trend suggests that the student-to-teacher trajectory mapping is relatively smooth and does not require deep hierarchical abstractions. A shallow and lightweight velocity field is therefore sufficient to model the feature transport, reinforcing the efficiency of our approach.

**Impact of velocity field hidden dimension.** We next investigate the effect of the velocity field width. Figure 5 (right) shows the ROUGE-L scores for velocity fields with varying hidden dimensions. For the GPT-2 setting (model dimension  $d \approx 768$ ), compact velocity fields with comparable dimensionality perform best. However, when applying CONTRA-KD to larger language models (e.g., LLaMA-2 13B  $\rightarrow$  7B, where hidden dimensions exceed 4096), we observe that narrow velocity fields introduce a severe *information bottleneck*. If the velocity field width is significantly smaller than the student or teacher hidden dimension, it fails to preserve the geometric structure of the feature manifold during projection, leading to

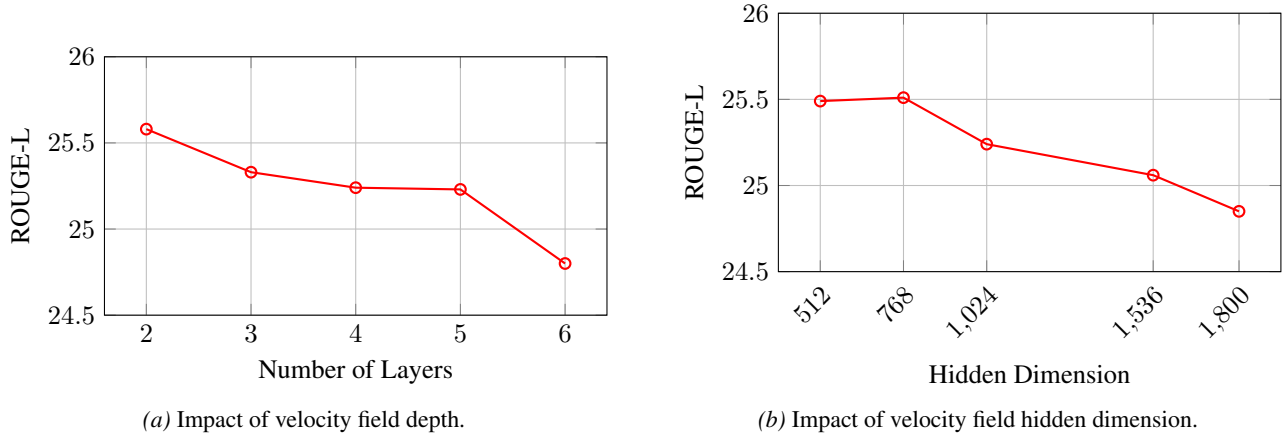


Figure 5. Ablation study on the architecture of the velocity field. **Left:** Increasing the depth of the velocity field degrades distillation performance, indicating that the student–teacher trajectory mapping is smooth and does not require deep hierarchies. **Right:** The velocity field width must scale with the base model dimension; overly narrow networks introduce an information bottleneck and reduce feature transport fidelity.

degraded performance. These results indicate that the velocity field dimension should scale proportionally with the base model size in order to maintain faithful feature transport.